Manual de referencia del lenguaje MQL 5

para el terminal de cliente MetaTrader 5

APRENDA el lenguaje MQL5 y REALICE cualquiera de estas tares:

- Creación de sus propios indicadores del análisis técnico de cualquier complejidad
- Autotrading – automatización del sistema comercial para operar en diferentes mercados financieros
- Diseño de herramientas analíticas a base de los avances matemáticos y métodos clásicos
- Desarrollo de sistemas informativo-comerciales para un amplio abanico de tareas (trading, monitoring, señales, etc.)

© 2000-2025, MetaQuotes Ltd

Contenido

Manual de referencia de MQL5 72

1 Bases del lenguaje ... 75

Sintaxis ... 76
Comentarios ... 77
Identificadores ... 78
Palabras reservadas ... 79
Tipos de datos ... 81
Tipos enteros ... 82
Tipos char, short, int y long ... 84
Constantes de caracteres ... 88
Tipo datetime ... 91
Tipo color ... 92
Tipo bool ... 93
Enumeraciones ... 94
Tipos reales (double, float) ... 96
Números complejos (complex) ... 102
Tipo string ... 103
Estructuras, clases e interfaces ... 106
Objeto de un array dinámico ... 133
Matrices y vectores ... 135
Conversión de tipos ... 143
Tipo void y constante NULL ... 148
Tipos personalizados ... 149
Punteros a objetos ... 159
Referencias Modificador & y palabra clave this ... 163
Operaciones y expresiones ... 165
Expresiones ... 166
Operaciones aritméticas ... 167
Operaciones de asignación ... 168
Operaciones de relación ... 169
Operaciones lógicas ... 170
Operaciones a nivel de bits ... 172
Otras operaciones ... 175
Prioridades y orden de las operaciones ... 179
Operadores ... 181
Operador compuesto ... 183
Operador-expresión ... 184
Operador de retorno return ... 185
Operador condicional if-else ... 186
Operador condicional ?: ... 187
Operador switch ... 189
Operador cíclico while ... 191
Operador cíclico for ... 192
Operador cíclico do while ... 194
Operador break ... 195
Operador de continuación continue ... 196
Operador de producto matricial ... 197
Operador-creador de objetos new ... 199
Operador-eliminador de objetos delete ... 201
Funciones ... 202
Llamada a una función ... 204
Traspaso de parámetros ... 205
Sobrecarga de funciones ... 208

© 2000-2025, MetaQuotes Ltd.

Sobrecarga de operaciones 211
Descripción de funciones externas 224
Exportación de funciones 226
Funciones de procesamiento de eventos 227
Variables 239
Variables locales 243
Parámetros formales 245
Variables estáticas 247
Variables globales 249
Variables Input 250
Variables Extern 257
Inicialización de variables 258
Visibilidad y tiempo de vida de variables 260
Creación y eliminación de objetos 262
Preprocesador 265
Declaración de constante (#define) 267
Propiedades de programas (#property) 270
Inclusión de archivos (#include) 276
Importación de funciones (#import) 277
Compilación condicional (#ifdef, #ifndef, #else, #endif) 280
Programación orientada a objetos 282
Encapsulación y extensión de tipos 284
Herencia 287
Polimorfismo 292
Sobrecarga 296
Funciones virtuales 297
Miembros estáticos de una clase 301
Plantillas de funciones 305
Plantillas de clases 309
Clases abstractas 314
Espacios de nombres 316
2 Constantes, Enumeraciones y Estructuras 320
Constantes de gráficos 321
Tipos de eventos de gráfico 322
Períodos de gráficos 330
Propiedades de gráficos 332
Posicionamiento de gráfico 346
Visualización de gráficos 347
Ejemplos de trabajo con el gráfico 349
Constantes de objetos 409
Tipos de objetos 410
OBJ_VLINE 412
OBJ_HLINE 417
OBJ_TREND 422
OBJ_TRENDBYANGLE 429
OBJ_CYCLES 435
OBJ_ARROWED_LINE 441
OBJ_CHANNEL 447
OBJ_STDDEVCHANNEL 454
OBJ_REGRESSION 461
OBJ_PITCHFORK 467
OBJ_GANNLINE 475
OBJ_GANNFAN 482
OBJ_GANNGRID 489
OBJ_FIBO 496
OBJ_FIBOTIMES 503
OBJ_FIBOFAN 510
OBJ_FIBOARC 517

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOCHANNEL ... 524
OBJ_EXPANSION ... 532
OBJ_ELLIOTWAVE5 ... 540
OBJ_ELLIOTWAVE3 ... 548
OBJ_RECTANGLE ... 555
OBJ_TRIANGLE ... 561
OBJ_ELLIPSE ... 568
OBJ_ARROW_THUMB_UP ... 574
OBJ_ARROW_THUMB_DOWN ... 580
OBJ_ARROW_UP ... 586
OBJ_ARROW_DOWN ... 592
OBJ_ARROW_STOP ... 598
OBJ_ARROW_CHECK ... 604
OBJ_ARROW_LEFT_PRICE ... 610
OBJ_ARROW_RIGHT_PRICE ... 615
OBJ_ARROW_BUY ... 620
OBJ_ARROW_SELL ... 625
OBJ_ARROW ... 630
OBJ_TEXT ... 636
OBJ_LABEL ... 642
OBJ_BUTTON ... 650
OBJ_CHART ... 657
OBJ_BITMAP ... 664
OBJ_BITMAP_LABEL ... 671
OBJ_EDIT ... 678
OBJ_EVENT ... 685
OBJ_RECTANGLE_LABEL ... 690
Propiedades de objetos ... 696
Modos de enlace de objetos ... 730
Esquina de enlace ... 735
Visibilidad de objetos ... 738
Niveles de las ondas de Elliott ... 741
Objetos de Gann ... 742
Colores Web ... 744
Wingdings ... 746
Constantes de indicadores ... 747
Constantes de precio ... 748
Métodos de alisamiento ... 751
Líneas de indicadores ... 752
Estilos de dibujo ... 754
Propiedades de indicadores personalizados ... 760
Tipos de indicadores ... 767
Identificadores de tipos de datos ... 769
Estado de entorno ... 770
Estado del terminal de cliente ... 771
Información sobre el programa MQL5 en ejecución ... 782
Información sobre el instrumento ... 787
Información sobre la cuenta ... 920
Estadística de simulación ... 930
Constantes comerciales ... 935
Información sobre datos históricos del instrumento ... 936
Propiedades de órdenes ... 937
Propiedades de posiciones ... 957
Propiedades de transacciones ... 962
Tipos de operaciones comerciales ... 967
Types of Trade Transactions ... 979
Tipos de órdenes en profundidad de mercado ... 982
Propiedades de señales ... 983

© 2000-2025, MetaQuotes Ltd.

Constantes nombradas **985**
Macro substituciones predefinidas 986
Constantes matemáticas 992
Constantes de tipos numéricos 994
Razones de deinicialización 997
Verificación del puntero a objeto 999
Otras constantes 1000
Estructuras de datos **1004**
Estructura de fecha 1005
Estructura de parámetros de entrada de indicador 1006
Estructura de datos históricos 1007
Estructura de profundidad de mercado 1008
Estructura de solicitud comercial 1009
Estructura de resultados de verificación de una solicitud comercial 1022
Estructura de resultado de solicitud comercial 1023
Structure of a Trade Transaction 1027
Estructura para obtención de precios actuales 1034
Estructuras del calendario económico 1036
Códigos de errores y advertencias **1045**
Códigos de retorno del servidor comercial 1046
Advertencias del compilador 1050
Errores de compilación 1053
Errores de tiempo de ejecución 1064
Constantes de entrada/salida **1079**
Banderas de apertura de archivos 1080
Propiedades de archivos 1082
Posicionamiento dentro del archivo 1083
Uso de página de código 1084
MessageBox 1085
3 Programas de MQL5 ..... **1087**
Ejecución de programas **1088**
Trade Permission **1096**
Eventos de terminal de cliente **1100**
Recursos **1103**
Llamadas a las funciones importadas **1115**
Errores de ejecución **1117**
Simulación de estrategias comerciales **1118**
4 Variables predefinidas **1145**
_AppliedTo **1146**
_Digits **1148**
_Point **1149**
_LastError **1150**
_Period **1151**
_RandomSeed **1152**
_StopFlag **1153**
_Symbol **1154**
_UninitReason **1155**
_IsX64 ..... **1156**
5 Funciones comunes **1157**
Alert **1159**
CheckPointer **1164**
Comment **1166**
CryptEncode **1168**
CryptDecode **1170**
DebugBreak **1172**
ExpertRemove **1173**
GetPointer **1175**

© 2000-2025, MetaQuotes Ltd.

GetTickCount ... 1178
GetTickCount64 ... 1179
GetMicrosecondCount ... 1181
MessageBox ... 1183
PeriodSeconds ... 1188
PlaySound ... 1190
Print ... 1193
PrintFormat ... 1195
ResetLastError ... 1201
ResourceCreate ... 1202
ResourceFree ... 1206
ResourceReadImage ... 1209
ResourceSave ... 1213
SetReturnError ... 1218
SetUserError ... 1220
Sleep ... 1221
TerminalClose ... 1223
TesterHideIndicators ... 1225
TesterStatistics ... 1227
TesterStop ... 1229
TesterDeposit ... 1231
TesterWithdrawal ... 1234
TranslateKey ... 1237
ZeroMemory ... 1238

6 Operaciones con arrays ... 1241

ArrayBsearch ... 1243
ArrayCopy ... 1247
ArrayCompare ... 1252
ArrayFree ... 1255
ArrayGetAsSeries ... 1264
ArrayInitialize ... 1267
ArrayFill ... 1269
ArrayIsDynamic ... 1271
ArrayIsSeries ... 1274
ArrayMaximum ... 1276
ArrayMinimum ... 1287
ArrayPrint ... 1298
ArrayRange ... 1301
ArrayResize ... 1302
ArrayInsert ... 1306
ArrayRemove ... 1309
ArrayReverse ... 1311
ArraySetAsSeries ... 1313
ArraySize ... 1315
ArraySort ... 1317
ArraySwap ... 1322
ArrayToFP16 ... 1324
ArrayToFP8 ... 1327
ArrayFromFP16 ... 1330
ArrayFromFP8 ... 1333

7 Métodos de matrices y vectores ... 1336

Tipos de matrices y vectores ... 1344
Enumeraciones ... 1346
Inicialización ... 1351
Assign ... 1354
CopyIndicatorBuffer ... 1356
CopyRates ... 1358
CopyTicks ... 1363

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange ... 1366
Eye ... 1368
Identity ... 1370
Ones ... 1372
Zeros ... 1373
Full ... 1374
Tri ... 1375
Init ... 1377
Fill ... 1379
Random ... 1380
Manipulación ... 1382
HasNan ... 1384
Transpose ... 1385
TransposeConjugate ... 1387
TriL ... 1388
TriU ... 1389
Diag ... 1390
Row ... 1392
Col ... 1394
Copy ... 1396
Concat ... 1398
Compare ... 1400
CompareByDigits ... 1402
CompareEqual ... 1404
Flat ... 1407
Clip ... 1409
Reshape ... 1410
Resize ... 1412
Set ... 1414
SwapRows ... 1416
SwapCols ... 1417
Split ... 1418
Hsplit ... 1420
Vsplit ... 1422
ArgSort ... 1424
Sort ... 1425
Operaciones ... 1427
Mathematical operations ... 1429
Mathematical functions ... 1430
Productos ... 1431
MatMul ... 1432
GeMM ... 1437
Power ... 1441
Dot ... 1444
Kron ... 1446
Inner ... 1448
Outer ... 1450
CorrCoef ... 1453
Cov ... 1457
Correlate ... 1460
Convolve ... 1463
Transformación ... 1466
Cholesky ... 1467
Eig ... 1468
EigVals ... 1471
LU ... 1472
LUP ... 1474
QR ... 1476

© 2000-2025, MetaQuotes Ltd.

SVD ... 1478
Estadísticas ... 1481
ArgMax ... 1482
ArgMin ... 1483
Max ... 1484
Min ... 1485
Ptp ... 1486
Sum ... 1487
Prod ... 1488
CumSum ... 1490
CumProd ... 1492
Percentile ... 1494
Quantile ... 1496
Median ... 1498
Mean ... 1500
Average ... 1501
Std ... 1503
Var ... 1505
LinearRegression ... 1507
Características ... 1510
Rows ... 1511
Cols ... 1512
Size ... 1513
Norm ... 1514
Cond ... 1517
Det ... 1520
SLogDet ... 1522
Rank ... 1523
Trace ... 1526
Spectrum ... 1527
Soluciones ... 1528
Solve ... 1529
LstSq ... 1530
Inv ... 1531
PInv ... 1533
Aprendizaje automático ... 1534
Activation ... 1540
Derivative ... 1544
Loss ... 1546
LossGradient ... 1548
RegressionMetric ... 1550
ConfusionMatrix ... 1552
ConfusionMatrixMultilabel ... 1554
ClassificationMetric ... 1556
ClassificationScore ... 1559
PrecisionRecall ... 1563
ReceiverOperatingCharacteristic ... 1567
OpenBLAS ... 1571
Singular Value Decomposition ... 1574
SingularValueDecompositionDC ... 1575
SingularValueDecompositionQR ... 1577
SingularValueDecompositionQRPivot ... 1579
SingularValueDecompositionBisect ... 1583
SingularValueDecompositionJacobiHigh ... 1586
SingularValueDecompositionJacobiLow ... 1591
SingularValueDecompositionBidiagDC ... 1594
SingularValueDecompositionBidiagBisect ... 1597
SingularValueDecompositionBidiagQR ... 1600

© 2000-2025, MetaQuotes Ltd.

Eigen Values ... 1602
General Matrices ... 1604
EigenSolver ... 1605
EigenSolverX ... 1607
EigenSolverShur ... 1611
EigenSolver2 ... 1613
EigenSolver2X ... 1616
EigenSolver2Shur ... 1621
EigenSolver2Blocked ... 1624
EigenSolver2ShurBlocked ... 1627
Symmetric Matrices ... 1630
EigenSymmetricDC ... 1632
EigenSymmetricQR ... 1634
EigenSymmetricRobust ... 1636
EigenSymmetricBisect ... 1639
EigenSymmetricDC2s ... 1642
EigenSymmetricQR2s ... 1644
EigenSymmetricRobust2s ... 1646
EigenSymmetricBisect2s ... 1649
EigenSymmetric2DC ... 1652
EigenSymmetric2QR ... 1655
EigenSymmetric2Bisect ... 1658
Tridiagonal Matrices ... 1662
EigenTridiagonalDC ... 1664
EigenTridiagonalQR ... 1666
EigenTridiagonalRobust ... 1668
EigenTridiagonalBisect ... 1671
EigenTridiagonalQL ... 1674
EigenTridiagonalDCQ ... 1675
EigenTridiagonalQRQ ... 1677
EigenTridiagonalPosDefQ ... 1679
Orthogonal Factorizations ... 1681
FactorizationQR ... 1682
FactorizationQRNonNeg ... 1684
FactorizationQRPivot ... 1686
FactorizationLQ ... 1688
FactorizationQL ... 1690
FactorizationRQ ... 1692
Linear Equations ... 1694
LinearEquationsSolution ... 1696
LinearEquationsSolutionTriangular ... 1698
LinearEquationsSolutionSy ... 1700
LinearEquationsSolutionComplexSy ... 1702
LinearEquationsSolutionSyPD ... 1703
LinearEquationsSolutionGeTrid ... 1705
LinearEquationsSolutionSyTridPD ... 1707
Factorizations ... 1709
FactorizationPLU ... 1710
FactorizationPLUGeTrid ... 1712
FactorizationLDL ... 1714
FactorizationLDLSyTridPD ... 1716
FactorizationCholesky ... 1718
FactorizationCholeskySyPS ... 1719
Dynamic Mode Decomposition ... 1721
DynamicModeDecomposition ... 1722
DynamicModeDecompositionQR ... 1728
Singular Spectrum Analysis ... 1735
SingularSpectrumAnalysisSpectrum ... 1736

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisForecast ... 1737
SingularSpectrumAnalysisReconstructComponents ... 1738
SingularSpectrumAnalysisReconstructSeries ... 1739
Reductions ... 1740
ReduceToBidiagonal ... 1741
ReflectBidiagonalToQP ... 1744
ReduceSymmetricToTridiagonal ... 1746
ReflectTridiagonalToQ ... 1748
8 Conversión de datos ... 1750
CharToString ... 1752
CharArrayToString ... 1754
CharArrayToStruct ... 1756
StructToCharArray ... 1760
ColorToARGB ... 1763
ColorToString ... 1765
DoubleToString ... 1766
EnumToString ... 1767
IntegerToString ... 1769
ShortToString ... 1771
ShortArrayToString ... 1773
TimeToString ... 1775
NormalizeDouble ... 1778
StringToCharArray ... 1780
StringToColor ... 1782
StringToDouble ... 1783
StringToInteger ... 1784
StringToShortArray ... 1785
StringToTime ... 1787
StringFormat ... 1790
9 Funciones matemáticas ... 1794
MathAbs ... 1796
MathArccos ... 1797
MathArcsin ... 1800
MathArctan ... 1803
MathArctan2 ... 1806
MathClassify ... 1810
MathCeil ... 1812
MathCos ... 1814
MathExp ... 1817
MathFloor ... 1820
MathLog ... 1822
MathLog10 ... 1826
MathMax ... 1830
MathMin ... 1832
MathMod ... 1834
MathPow ... 1835
MathRand ... 1839
MathRound ... 1840
MathSin ... 1842
MathSqrt ... 1845
MathSrand ... 1849
MathTan ... 1852
MathIsValidNumber ... 1855
MathExpm1 ... 1856
MathLog1p ... 1859
MathArccosh ... 1863
MathArcsinh ... 1866
MathArctanh ... 1869

© 2000-2025, MetaQuotes Ltd.

MathCosh 1872
MathSinh 1875
MathTanh 1878
MathSwap 1881
10 Funciones de cadenas de caracteres 1884
StringAdd 1885
StringBufferLen 1887
StringCompare 1888
StringConcatenate 1890
StringFill 1892
StringFind 1893
StringGetCharacter 1896
StringInit 1898
StringLen 1899
StringSetLength 1900
StringReplace 1901
StringReserve 1902
StringSetCharacter 1904
StringSplit 1906
StringSubstr 1907
StringToLower 1908
StringToUpper 1909
StringTrimLeft 1910
StringTrimRight 1911
11 Fecha y hora 1912
TimeCurrent 1913
TimeTradeServer 1915
TimeLocal 1917
TimeGMT 1919
TimeDaylightSavings 1921
TimeGMTOffset 1922
TimeToStruct 1923
StructToTime 1924
12 Información de cuenta 1926
AccountInfoDouble 1927
AccountInfoInteger 1928
AccountInfoString 1930
13 Comprobación de estado 1931
GetLastError 1932
IsStopped 1933
UninitializeReason 1934
TerminalInfoInteger 1936
TerminalInfoDouble 1937
TerminalInfoString 1938
MQLInfoInteger 1939
MQLInfoString 1940
Symbol 1941
Period 1942
Digits 1943
Point 1944
14 Procesamiento de eventos 1945
OnStart 1947
OnInit 1950
OnDeinit 1953
OnTick 1956
OnCalculate 1962

© 2000-2025, MetaQuotes Ltd.

OnTimer 1966
OnTrade 1969
OnTradeTransaction 1974
OnBookEvent 1980
OnChartEvent 1984
OnTester 1991
OnTesterInit 1998
OnTesterDeinit 2005
OnTesterPass 2006
15 Obtención de información de mercado 2007
SymbolsTotal 2008
SymbolExist 2009
SymbolName 2011
SymbolSelect 2013
SymbolIsSynchronized 2015
SymbolInfoDouble 2016
SymbolInfoInteger 2018
SymbolInfoString 2020
SymbolInfoMarginRate 2022
SymbolInfoTick 2024
SymbolInfoSessionQuote 2025
SymbolInfoSessionTrade 2027
MarketBookAdd 2029
MarketBookRelease 2031
MarketBookGet 2033
16 Calendario económico 2034
CalendarCountryById 2035
CalendarEventById 2037
CalendarValueById 2040
CalendarCountries 2043
CalendarEventByCountry 2045
CalendarEventByCurrency 2047
CalendarValueHistoryByEvent 2049
CalendarValueHistory 2053
CalendarValueLastByEvent 2056
CalendarValueLast 2061
17 Acceso a las series temporales y a los datos de indicadores 2066
Dirección de indexación en los arrays y series temporales 2071
Organización de acceso a los datos 2075
SeriesInfoInteger 2085
Bars 2087
BarsCalculated 2090
IndicatorCreate 2092
IndicatorParameters 2094
IndicatorRelease 2096
CopyBuffer 2098
CopyRates 2103
CopySeries 2107
CopyTime 2111
CopyOpen 2114
CopyHigh 2117
CopyLow 2121
CopyClose 2124
CopyTickVolume 2127
CopyRealVolume 2131
CopySpread 2134
CopyTicks 2138

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange 2144
iBars 2146
iBarShift 2147
iClose 2150
iHigh 2152
iHighest 2154
iLow 2155
iLowest 2157
iOpen 2158
iTime 2160
iTickVolume 2162
iRealVolume 2164
iVolume 2166
iSpread 2168

18 Símbolos personalizados 2170

CustomSymbolCreate 2172
CustomSymbolDelete 2175
CustomSymbolSetInteger 2177
CustomSymbolSetDouble 2181
CustomSymbolSetString 2185
CustomSymbolSetMarginRate 2189
CustomSymbolSetSessionQuote 2194
CustomSymbolSetSessionTrade 2199
CustomRatesDelete 2204
CustomRatesReplace 2209
CustomRatesUpdate 2214
CustomTicksAdd 2219
CustomTicksDelete 2227
CustomTicksReplace 2235
CustomBookAdd 2244

19 Operaciones con gráficos 2247

ChartApplyTemplate 2249
ChartSaveTemplate 2252
ChartWindowFind 2257
ChartTimePriceToXY 2259
ChartXYToTimePrice 2261
ChartOpen 2263
ChartFirst 2265
ChartNext 2266
ChartClose 2267
ChartSymbol 2269
ChartPeriod 2270
ChartRedraw 2271
ChartSetDouble 2274
ChartSetInteger 2276
ChartSetString 2278
ChartGetDouble 2280
ChartGetInteger 2282
ChartGetString 2284
ChartNavigate 2286
ChartID 2289
ChartIndicatorAdd 2291
ChartIndicatorDelete 2294
ChartIndicatorGet 2297
ChartIndicatorName 2299
ChartIndicatorsTotal 2301
ChartWindowOnDropped 2303
ChartPriceOnDropped 2304

© 2000-2025, MetaQuotes Ltd.

ChartTimeOnDropped 2305
ChartXOnDropped 2306
ChartYOnDropped 2307
ChartSetSymbolPeriod 2308
ChartScreenShot 2310
20 Funciones comerciales 2313
OrderCalcMargin 2316
OrderCalcProfit 2320
OrderCheck 2323
OrderSend 2327
OrderSendAsync 2332
PositionsTotal 2343
PositionGetSymbol 2344
PositionSelect 2346
PositionSelectByTicket 2348
PositionGetDouble 2350
PositionGetInteger 2353
PositionGetString 2355
PositionGetTicket 2358
OrdersTotal 2360
OrderGetTicket 2361
OrderSelect 2363
OrderGetDouble 2367
OrderGetInteger 2371
OrderGetString 2377
HistorySelect 2380
HistorySelectByPosition 2382
HistoryOrderSelect 2386
HistoryOrdersTotal 2389
HistoryOrderGetTicket 2390
HistoryOrderGetDouble 2392
HistoryOrderGetInteger 2396
HistoryOrderGetString 2399
HistoryDealSelect 2402
HistoryDealsTotal 2405
HistoryDealGetTicket 2406
HistoryDealGetDouble 2409
HistoryDealGetInteger 2414
HistoryDealGetString 2417
21 Administrar señales 2421
SignalBaseGetDouble 2422
SignalBaseGetInteger 2423
SignalBaseGetString 2424
SignalBaseSelect 2425
SignalBaseTotal 2426
SignalInfoGetDouble 2427
SignalInfoGetInteger 2428
SignalInfoGetString 2429
SignalInfoSetDouble 2430
SignalInfoSetInteger 2431
SignalSubscribe 2432
SignalUnsubscribe 2433
22 Funciones de red 2434
SocketCreate 2436
SocketClose 2439
SocketConnect 2442
SocketIsConnected 2446

© 2000-2025, MetaQuotes Ltd.

SocketIsReadable 2449
SocketIsWritable 2452
SocketTimeouts 2456
SocketRead 2458
SocketSend 2462
SocketTlsHandshake 2466
SocketTlsCertificate 2471
SocketTlsRead 2475
SocketTlsReadAvailable 2479
SocketTlsSend 2484
WebRequest 2488
SendFTP 2491
SendMail 2493
SendNotification 2495
23 Variables globales del terminal de cliente 2497
GlobalVariableCheck 2498
GlobalVariableTime 2499
GlobalVariableDel 2500
GlobalVariableGet 2501
GlobalVariableName 2502
GlobalVariableSet 2503
GlobalVariablesFlush 2504
GlobalVariableTemp 2505
GlobalVariableSetOnCondition 2506
GlobalVariablesDeleteAll 2507
GlobalVariablesTotal 2508
24 Operaciones con archivos 2509
FileSelectDialog 2512
FileFindFirst 2515
FileFindNext 2517
FileFindClose 2519
FileIsExist 2520
FileOpen 2523
FileClose 2526
FileCopy 2527
FileDelete 2530
FileMove 2533
FileFlush 2535
FileGetInteger 2537
FileIsEnding 2540
FileIsLineEnding 2541
FileReadArray 2547
FileReadBool 2549
FileReadDatetime 2553
FileReadDouble 2557
FileReadFloat 2560
FileReadInteger 2564
FileReadLong 2568
FileReadNumber 2571
FileReadString 2577
FileReadStruct 2579
FileSeek 2583
FileSize 2586
FileTell 2589
FileWrite 2592
FileWriteArray 2595
FileWriteDouble 2598
FileWriteFloat 2601

© 2000-2025, MetaQuotes Ltd.

FileWriteInteger 2604
FileWriteLong 2607
FileWriteString 2610
FileWriteStruct 2613
FileLoad 2616
FileSave 2618
FolderCreate 2620
FolderDelete 2623
FolderClean 2626
25 Indicadores personalizados 2629
Estilos de indicadores en ejemplos 2633
DRAW_NONE 2645
DRAW_LINE 2648
DRAW_SECTION 2652
DRAW_HISTOGRAM 2656
DRAW_HISTOGRAM2 2660
DRAW_ARROW 2664
DRAW_ZIGZAG 2669
DRAW_FILLING 2674
DRAW_BARS 2679
DRAW_CANDLES 2685
DRAW_COLOR_LINE 2692
DRAW_COLOR_SECTION 2697
DRAW_COLOR_HISTOGRAM 2703
DRAW_COLOR_HISTOGRAM2 2708
DRAW_COLOR_ARROW 2713
DRAW_COLOR_ZIGZAG 2719
DRAW_COLOR_BARS 2724
DRAW_COLOR_CANDLES 2731
Relación entre propiedades de indicador y funciones 2738
SetIndexBuffer 2741
IndicatorSetDouble 2744
IndicatorSetInteger 2748
IndicatorSetString 2752
PlotIndexSetDouble 2755
PlotIndexSetInteger 2756
PlotIndexSetString 2760
PlotIndexGetInteger 2761
26 Objetos gráficos 2764
ObjectCreate 2766
ObjectName 2770
ObjectDelete 2771
ObjectsDeleteAll 2772
ObjectFind 2773
ObjectGetTimeByValue 2774
ObjectGetValueByTime 2775
ObjectMove 2776
ObjectsTotal 2777
ObjectSetDouble 2778
ObjectSetInteger 2782
ObjectSetString 2786
ObjectGetDouble 2788
ObjectGetInteger 2790
ObjectGetString 2792
TextSetFont 2794
TextOut 2797
TextGetSize 2801

© 2000-2025, MetaQuotes Ltd.

27 Indicadores técnicos 2802
iAC 2805
iAD 2810
iADX 2815
iADXWilder 2820
iAlligator 2825
iAMA 2832
iAO 2837
iATR 2842
iBearsPower 2847
iBands 2852
iBullsPower 2858
iCCI 2863
iChaikin 2868
iCustom 2873
iDEMA 2877
iDeMarker 2882
iEnvelopes 2887
iForce 2893
iFractals 2898
iFrAMA 2903
iGator 2908
Ilchimoku 2915
iBWMFI 2922
iMomentum 2927
iMFI 2932
iMA 2937
iOsMA 2942
iMACD 2947
iOBV 2953
iSAR 2958
iRSI 2963
iRVI 2968
iStdDev 2973
iStochastic 2978
iTEMA 2984
iTriX 2989
iWPR 2994
iVIDyA 2999
iVolumes 3004
28 Trabajo con resultados de optimización 3009
FrameFirst 3011
FrameFilter 3012
FrameNext 3013
FrameInputs 3014
FrameAdd 3015
ParameterGetRange 3016
ParameterSetRange 3020
29 Trabajo con eventos 3022
EventSetMillisecondTimer 3023
EventSetTimer 3024
EventKillTimer 3025
EventChartCustom 3026
30 Trabajo con OpenCL 3032
CLHandleType 3034
CLGetInfoInteger 3035
CLGetInfoString 3038

© 2000-2025, MetaQuotes Ltd.

CLContextCreate 3041
CLContextFree 3042
CLGetDeviceInfo 3043
CLProgramCreate 3048
CLProgramFree 3053
CLKernelCreate 3054
CLKernelFree 3055
CLSetKernelArg 3056
CLSetKernelArgMem 3057
CLSetKernelArgMemLocal 3058
CLBufferCreate 3059
CLBufferFree 3060
CLBufferWrite 3061
CLBufferRead 3066
CLExecute 3070
CLExecutionStatus 3072

31 Trabajo con la base de datos 3073

DatabaseOpen 3077
DatabaseClose 3079
DatabaseImport 3080
DatabaseExport 3083
DatabasePrint 3089
DatabaseTableExists 3094
DatabaseExecute 3095
DatabasePrepare 3107
DatabaseReset 3116
DatabaseBind 3122
DatabaseBindArray 3127
DatabaseRead 3132
DatabaseReadBind 3133
DatabaseFinalize 3137
DatabaseTransactionBegin 3138
DatabaseTransactionCommit 3143
DatabaseTransactionRollback 3144
DatabaseColumnsCount 3145
DatabaseColumnName 3146
DatabaseColumnType 3147
DatabaseColumnSize 3148
DatabaseColumnText 3149
DatabaseColumnInteger 3150
DatabaseColumnLong 3151
DatabaseColumnDouble 3152
DatabaseColumnBlob 3153

32 Trabajo con DirectX 3154

DXContextCreate 3156
DXContextSetSize 3157
DXContextGetSize 3158
DXContextClearColors 3159
DXContextClearDepth 3160
DXContextGetColors 3161
DXContextGetDepth 3162
DXBufferCreate 3163
DXTextureCreate 3164
DXInputCreate 3170
DXInputSet 3171
DXShaderCreate 3172
DXShaderSetLayout 3173
DXShaderInputsSet 3174

© 2000-2025, MetaQuotes Ltd.

DXShaderTexturesSet 3175
DXDraw 3176
DXDrawIndexed 3177
DXPrimiveTopologySet 3178
DXBufferSet 3179
DXShaderSet 3180
DXHandleType 3181
DXRelease 3182
33 MetaTrader para Python 3183
initialize 3189
login 3191
shutdown 3194
version 3195
last_error 3197
account_info 3199
terminal_info 3202
symbols_total 3205
symbols_get 3206
symbol_info 3209
symbol_info_tick 3213
symbol_select 3215
market_book_add 3219
market_book_get 3220
market_book_release 3223
copy_rates_from 3224
copy_rates_from_pPos 3228
copy_rates_range 3231
copy_ticks_from 3234
copy_ticks_range 3238
orders_total 3241
orders_get 3242
order_calc_margin 3245
order_calc_profit 3248
order_check 3251
order_send 3255
positions_total 3260
positions_get 3261
history_orders_total 3264
history_orders_get 3266
history_deals_total 3269
history_deals_get 3271
34 Modelos ONNX 3275
Soporte de ONNX 3276
Conversión de formatos 3278
Autoconversión de los datos 3279
Creación de un modelo 3283
Inicio del modelo 3291
Comprobación en el simulador 3297
OnnxCreate 3302
OnnxCreateFromBuffer 3303
OnnxRelease 3304
OnnxRun 3305
OnnxGetInputCount 3308
OnnxGetOutputCount 3309
OnnxGetInputName 3310
OnnxGetOutputName 3311
OnnxGetInputTypeInfo 3312
OnnxGetOutputTypeInfo 3313

© 2000-2025, MetaQuotes Ltd.

OnnxSetInputShape 3314
OnnxSetOutputShape 3316
Estructura de los datos 3318
35 Biblioteca estándar 3322
Matemáticas 3323
Estadística 3324
Características estadísticas 3327
MathMean 3328
MathVariance 3329
MathSkewness 3330
MathKurtosis 3331
MathMoments 3332
MathMedian 3333
MathStandardDeviation 3334
MathAverageDeviation 3335
Distribución normal 3336
MathProbabilityDensityNormal 3340
MathCumulativeDistributionNormal 3342
MathQuantileNormal 3344
MathRandomNormal 3346
MathMomentsNormal 3347
Distribución log-normal 3348
MathProbabilityDensityLognormal 3352
MathCumulativeDistributionLognormal 3354
MathQuantileLognormal 3356
MathRandomLognormal 3358
MathMomentsLognormal 3359
Distribución beta 3360
MathProbabilityDensityBeta 3364
MathCumulativeDistributionBeta 3366
MathQuantileBeta 3368
MathRandomBeta 3370
MathMomentsBeta 3371
Distribución beta no central 3372
MathProbabilityDensityNoncentralBeta 3376
MathCumulativeDistributionNoncentralBeta 3378
MathQuantileNoncentralBeta 3380
MathRandomNoncentralBeta 3382
MathMomentsNoncentralBeta 3383
Distribución gamma 3384
MathProbabilityDensityGamma 3388
MathCumulativeDistributionGamma 3390
MathQuantileGamma 3392
MathRandomGamma 3394
MathMomentsGamma 3395
Distribución chi-cuadrado 3396
MathProbabilityDensityChiSquare 3400
MathCumulativeDistributionChiSquare 3402
MathQuantileChiSquare 3404
MathRandomChiSquare 3406
MathMomentsChiSquare 3407
Distribución chi-cuadrado no central 3408
MathProbabilityDensityNoncentralChiSquare 3412
MathCumulativeDistributionNoncentralChiSquare 3414
MathQuantileNoncentralChiSquare 3416
MathRandomNoncentralChiSquare 3418
MathMomentsNoncentralChiSquare 3419
Distribución exponencial 3420

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityExponential ... 3424
MathCumulativeDistributionExponential ... 3426
MathQuantileExponential ... 3428
MathRandomExponential ... 3430
MathMomentsExponential ... 3431
Distribución F ... 3432
MathProbabilityDensityF ... 3436
MathCumulativeDistributionF ... 3438
MathQuantileF ... 3440
MathRandomF ... 3442
MathMomentsF ... 3443
Distribución F no central ... 3444
MathProbabilityDensityNoncentralF ... 3448
MathCumulativeDistributionNoncentralF ... 3450
MathQuantileNoncentralF ... 3452
MathRandomNoncentralF ... 3454
MathMomentsNoncentralF ... 3455
Distribución T ... 3456
MathProbabilityDensityT ... 3460
MathCumulativeDistributionT ... 3462
MathQuantileT ... 3464
MathRandomT ... 3466
MathMomentsT ... 3467
Distribución T no central ... 3468
MathProbabilityDensityNoncentralT ... 3472
MathCumulativeDistributionNoncentralT ... 3474
MathQuantileNoncentralT ... 3476
MathRandomNoncentralT ... 3478
MathMomentsNoncentralT ... 3479
Distribución logística ... 3480
MathProbabilityDensityLogistic ... 3484
MathCumulativeDistributionLogistic ... 3486
MathQuantileLogistic ... 3488
MathRandomLogistic ... 3490
MathMomentsLogistic ... 3491
Distribución de Cauchy ... 3492
MathProbabilityDensityCauchy ... 3496
MathCumulativeDistributionCauchy ... 3498
MathQuantileCauchy ... 3500
MathRandomCauchy ... 3502
MathMomentsCauchy ... 3503
Distribución uniforme ... 3504
MathProbabilityDensityUniform ... 3508
MathCumulativeDistributionUniform ... 3510
MathQuantileUniform ... 3512
MathRandomUniform ... 3514
MathMomentsUniform ... 3515
Distribución de Weibull ... 3516
MathProbabilityDensityWeibull ... 3520
MathCumulativeDistributionWeibull ... 3522
MathQuantileWeibull ... 3524
MathRandomWeibull ... 3526
MathMomentsWeibull ... 3527
Distribución binomial ... 3528
MathProbabilityDensityBinomial ... 3531
MathCumulativeDistributionBinomial ... 3533
MathQuantileBinomial ... 3535
MathRandomBinomial ... 3537

© 2000-2025, MetaQuotes Ltd.

MathMomentsBinomial ... 3538
Distribución binomial negativa ... 3539
MathProbabilityDensityNegativeBinomial ... 3542
MathCumulativeDistributionNegativeBinomial ... 3544
MathQuantileNegativeBinomial ... 3546
MathRandomNegativeBinomial ... 3548
MathMomentsNegativeBinomial ... 3549
Distribución geométrica ... 3550
MathProbabilityDensityGeometric ... 3554
MathCumulativeDistributionGeometric ... 3556
MathQuantileGeometric ... 3558
MathRandomGeometric ... 3560
MathMomentsGeometric ... 3561
Distribución hipergeométrica ... 3562
MathProbabilityDensityHypergeometric ... 3566
MathCumulativeDistributionHypergeometric ... 3568
MathQuantileHypergeometric ... 3570
MathRandomHypergeometric ... 3572
MathMomentsHypergeometric ... 3573
Distribución de Poisson ... 3574
MathProbabilityDensityPoisson ... 3578
MathCumulativeDistributionPoisson ... 3580
MathQuantilePoisson ... 3582
MathRandomPoisson ... 3584
MathMomentsPoisson ... 3585
Funciones auxiliares ... 3586
MathRandomNonZero ... 3591
MathMoments ... 3592
MathPowInt ... 3593
MathFactorial ... 3594
MathTrunc ... 3595
MathRound ... 3596
MathArctan2 ... 3598
MathGamma ... 3600
MathGammaLog ... 3601
MathBeta ... 3602
MathBetaLog ... 3603
MathBetaIncomplete ... 3604
MathGammaIncomplete ... 3605
MathBinomialCoefficient ... 3606
MathBinomialCoefficientLog ... 3607
MathHypergeometric2F2 ... 3608
MathSequence ... 3609
MathSequenceByCount ... 3610
MathReplicate ... 3611
MathReverse ... 3612
MathIdentical ... 3613
MathUnique ... 3614
MathQuickSortAscending ... 3615
MathQuickSortDescending ... 3616
MathQuickSort ... 3617
MathOrder ... 3618
MathBitwiseNot ... 3619
MathBitwiseAnd ... 3620
MathBitwiseOr ... 3621
MathBitwiseXor ... 3622
MathBitwiseShiftL ... 3623
MathBitwiseShiftR ... 3624

© 2000-2025, MetaQuotes Ltd.

MathCumulativeSum ... 3625
MathCumulativeProduct ... 3626
MathCumulativeMin ... 3627
MathCumulativeMax ... 3628
MathSin ... 3629
MathCos ... 3630
MathTan ... 3631
MathArcsin ... 3632
MathArccos ... 3633
MathArctan ... 3634
MathSinPi ... 3635
MathCosPi ... 3636
MathTanPi ... 3637
MathAbs ... 3638
MathCeil ... 3639
MathFloor ... 3640
MathSqrt ... 3641
MathExp ... 3642
MathPow ... 3643
MathLog ... 3644
MathLog2 ... 3645
MathLog10 ... 3646
MathLog1p ... 3647
MathDifference ... 3648
MathSample ... 3650
MathTukeySummary ... 3653
MathRange ... 3654
MathMin ... 3655
MathMax ... 3656
MathSum ... 3657
MathProduct ... 3658
MathStandardDeviation ... 3659
MathAverageDeviation ... 3660
MathMedian ... 3661
MathMean ... 3662
MathVariance ... 3663
MathSkewness ... 3664
MathKurtosis ... 3665
MathExpm1 ... 3666
MathSinh ... 3667
MathCosh ... 3668
MathTanh ... 3669
MathArcsinh ... 3670
MathArccosh ... 3671
MathArctanh ... 3672
MathSignif ... 3673
MathRank ... 3675
MathCorrelationPearson ... 3676
MathCorrelationSpearman ... 3677
MathCorrelationKendall ... 3678
MathQuantile ... 3679
MathProbabilityDensityEmpirical ... 3680
MathCumulativeDistributionEmpirical ... 3681
Lógica difusa ... 3682
Funciones de pertenencia ... 3683
CConstantMembershipFunction ... 3685
GetValue ... 3687
CCompositeMembershipFunction ... 3688

© 2000-2025, MetaQuotes Ltd.

CompositionType ... 3690
MembershipFunctions ... 3690
GetValue ... 3690
CDifferencTwoSigmoidalMembershipFunction ... 3692
A1 ... 3694
A2 ... 3694
C1 ... 3695
C2 ... 3695
GetValue ... 3696
CGeneralizedBellShapedMembershipFunction ... 3697
A ... 3699
B ... 3699
C ... 3700
GetValue ... 3700
CNormalCombinationMembershipFunction ... 3701
B1 ... 3703
B2 ... 3703
Sigma1 ... 3704
Sigma2 ... 3704
GetValue ... 3705
CNormalMembershipFunction ... 3706
B ... 3708
Sigma ... 3708
GetValue ... 3709
CP_ShapedMembershipFunction ... 3710
A ... 3712
B ... 3712
C ... 3713
D ... 3713
GetValue ... 3714
CProductTwoSigmoidalMembershipFunctions ... 3715
A1 ... 3717
A2 ... 3717
C1 ... 3718
C2 ... 3718
GetValue ... 3719
CS_ShapedMembershipFunction ... 3720
A ... 3722
B ... 3722
GetValue ... 3723
CSigmoidalMembershipFunction ... 3724
A ... 3726
C ... 3726
GetValue ... 3727
CTrapezoidMembershipFunction ... 3728
X1 ... 3730
X2 ... 3730
X3 ... 3731
X4 ... 3731
GetValue ... 3732
CTriangularMembershipFunction ... 3733
X1 ... 3735
X2 ... 3735
X3 ... 3736
ToNormalMF ... 3736
GetValue ... 3736
CZ_ShapedMembershipFunction ... 3738
A ... 3740

© 2000-2025, MetaQuotes Ltd.

B ... 3740
GetValue ... 3741
IMembershipFunction ... 3742
GetValue ... 3742
Reglas de los sistemas difusos ... 3743
CMamdaniFuzzyRule ... 3744
Conclusion ... 3745
Weight ... 3745
CSugenoFuzzyRule ... 3746
Conclusion ... 3747
CSingleCondition ... 3748
Not ... 3748
Term ... 3749
Var ... 3749
CConditions ... 3751
ConditionsList ... 3751
Not ... 3752
Op ... 3752
CGenericFuzzyRule ... 3754
Conclusion ... 3754
Condition ... 3755
CreateCondition ... 3755
Variables para los sistemas difusos ... 3757
CFuzzyVariable ... 3758
AddTerm ... 3759
GetTermByName ... 3759
Max ... 3759
Min ... 3760
Terms ... 3760
Values ... 3761
CSugenoVariable ... 3762
Functions ... 3762
GetFuncByName ... 3763
Values ... 3763
Términos difusos ... 3764
MembershipFunction ... 3765
Sistemas difusos ... 3766
Sistema Mamdani ... 3767
AggregationMethod ... 3767
Calculate ... 3768
DefuzzificationMethod ... 3768
EmptyRule ... 3768
ImplicationMethod ... 3768
Output ... 3769
OutputByName ... 3769
ParseRule ... 3769
Rules ... 3770
Sistema Sugeno ... 3771
Calculate ... 3771
CreateSugenoFunction ... 3772
EmptyRule ... 3773
Output ... 3773
OutputByName ... 3773
ParseRule ... 3773
Rules ... 3774
OpenCL ... 3775
BufferCreate ... 3777
BufferFree ... 3778

© 2000-2025, MetaQuotes Ltd.

BufferFromArray ... 3779
BufferRead ... 3780
BufferWrite ... 3781
Execute ... 3782
GetContext ... 3783
GetKernel ... 3784
GetKernelName ... 3785
GetProgram ... 3786
Initialize ... 3787
KernelCreate ... 3788
KernelFree ... 3789
SetArgument ... 3790
SetArgumentBuffer ... 3791
SetArgumentLocalMemory ... 3792
SetBuffersCount ... 3793
SetKernelsCount ... 3794
Shutdown ... 3795
SupportDouble ... 3796
Clase base CObject ... 3797
Prev ... 3798
Prev ... 3799
Next ... 3800
Next ... 3801
Compare ... 3802
Save ... 3804
Load ... 3806
Type ... 3808
Colecciones de datos ... 3809
CArray ... 3810
Step ... 3812
Step ... 3813
Total ... 3814
Available ... 3815
Max ... 3816
IsSorted ... 3817
SortMode ... 3818
Clear ... 3819
Sort ... 3820
Save ... 3821
Load ... 3822
CArrayChar ... 3823
Reserve ... 3826
Resize ... 3827
Shutdown ... 3828
Add ... 3829
AddArray ... 3830
AddArray ... 3831
Insert ... 3833
InsertArray ... 3834
InsertArray ... 3835
AssignArray ... 3837
AssignArray ... 3838
Update ... 3840
Shift ... 3841
Delete ... 3842
DeleteRange ... 3843
At ... 3844
CompareArray ... 3846

© 2000-2025, MetaQuotes Ltd.

CompareArray ... 3847
InsertSort ... 3848
Search ... 3849
SearchGreat ... 3850
SearchLess ... 3851
SearchGreatOrEqual ... 3852
SearchLessOrEqual ... 3853
SearchFirst ... 3854
SearchLast ... 3855
SearchLinear ... 3856
Save ... 3857
Load ... 3858
Type ... 3860
CArrayShort ... 3861
Reserve ... 3863
Resize ... 3864
Shutdown ... 3865
Add ... 3866
AddArray ... 3867
AddArray ... 3868
Insert ... 3870
InsertArray ... 3871
InsertArray ... 3872
AssignArray ... 3874
AssignArray ... 3875
Update ... 3877
Shift ... 3878
Delete ... 3879
DeleteRange ... 3880
At ... 3881
CompareArray ... 3883
CompareArray ... 3884
InsertSort ... 3885
Search ... 3886
SearchGreat ... 3887
SearchLess ... 3888
SearchGreatOrEqual ... 3889
SearchLessOrEqual ... 3890
SearchFirst ... 3891
SearchLast ... 3892
SearchLinear ... 3893
Save ... 3894
Load ... 3896
Type ... 3898
CArrayInt ... 3899
Reserve ... 3902
Resize ... 3903
Shutdown ... 3904
Add ... 3905
AddArray ... 3906
AddArray ... 3907
Insert ... 3909
InsertArray ... 3910
InsertArray ... 3911
AssignArray ... 3913
AssignArray ... 3914
Update ... 3916
Shift ... 3917

© 2000-2025, MetaQuotes Ltd.

Delete ... 3918
DeleteRange ... 3919
At ... 3920
CompareArray ... 3922
CompareArray ... 3923
InsertSort ... 3924
Search ... 3925
SearchGreat ... 3926
SearchLess ... 3927
SearchGreatOrEqual ... 3928
SearchLessOrEqual ... 3929
SearchFirst ... 3930
SearchLast ... 3931
SearchLinear ... 3932
Save ... 3933
Load ... 3935
Type ... 3937
CArrayLong ... 3938
Reserve ... 3941
Resize ... 3942
Shutdown ... 3943
Add ... 3944
AddArray ... 3945
AddArray ... 3946
Insert ... 3948
InsertArray ... 3949
InsertArray ... 3950
AssignArray ... 3952
AssignArray ... 3953
Update ... 3955
Shift ... 3956
Delete ... 3957
DeleteRange ... 3958
At ... 3959
CompareArray ... 3961
CompareArray ... 3962
InsertSort ... 3963
Search ... 3964
SearchGreat ... 3965
SearchLess ... 3966
SearchGreatOrEqual ... 3967
SearchLessOrEqual ... 3968
SearchFirst ... 3969
SearchLast ... 3970
SearchLinear ... 3971
Save ... 3972
Load ... 3974
Type ... 3976
CArrayFloat ... 3977
Delta ... 3980
Reserve ... 3981
Resize ... 3982
Shutdown ... 3983
Add ... 3984
AddArray ... 3985
AddArray ... 3986
Insert ... 3988
InsertArray ... 3989

© 2000-2025, MetaQuotes Ltd.

InsertArray ... 3990
AssignArray ... 3992
AssignArray ... 3993
Update ... 3995
Shift ... 3996
Delete ... 3997
DeleteRange ... 3998
At ... 3999
CompareArray ... 4001
CompareArray ... 4002
InsertSort ... 4003
Search ... 4004
SearchGreat ... 4005
SearchLess ... 4006
SearchGreatOrEqual ... 4007
SearchLessOrEqual ... 4008
SearchFirst ... 4009
SearchLast ... 4010
SearchLinear ... 4011
Save ... 4012
Load ... 4014
Type ... 4016
CArrayDouble ... 4017
Delta ... 4020
Reserve ... 4021
Resize ... 4022
Shutdown ... 4023
Add ... 4024
AddArray ... 4025
AddArray ... 4026
Insert ... 4028
InsertArray ... 4029
InsertArray ... 4030
AssignArray ... 4032
AssignArray ... 4033
Update ... 4035
Shift ... 4036
Delete ... 4037
DeleteRange ... 4038
At ... 4039
CompareArray ... 4041
CompareArray ... 4042
Minimum ... 4043
Maximum ... 4044
InsertSort ... 4045
Search ... 4046
SearchGreat ... 4047
SearchLess ... 4048
SearchGreatOrEqual ... 4049
SearchLessOrEqual ... 4050
SearchFirst ... 4051
SearchLast ... 4052
SearchLinear ... 4053
Save ... 4054
Load ... 4056
Type ... 4058
CArrayString ... 4059
Reserve ... 4061

© 2000-2025, MetaQuotes Ltd.

Resize ... 4062
Shutdown ... 4063
Add ... 4064
AddArray ... 4065
AddArray ... 4066
Insert ... 4068
InsertArray ... 4069
InsertArray ... 4070
AssignArray ... 4072
AssignArray ... 4073
Update ... 4075
Shift ... 4076
Delete ... 4077
DeleteRange ... 4078
At ... 4079
CompareArray ... 4081
CompareArray ... 4082
InsertSort ... 4083
Search ... 4084
SearchGreat ... 4085
SearchLess ... 4086
SearchGreatOrEqual ... 4087
SearchLessOrEqual ... 4088
SearchFirst ... 4089
SearchLast ... 4090
SearchLinear ... 4091
Save ... 4092
Load ... 4094
Type ... 4096
CArrayObj ... 4097
FreeMode ... 4102
FreeMode ... 4103
Reserve ... 4105
Resize ... 4106
Clear ... 4107
Shutdown ... 4108
CreateElement ... 4109
Add ... 4111
AddArray ... 4112
Insert ... 4115
InsertArray ... 4117
AssignArray ... 4119
Update ... 4121
Shift ... 4122
Detach ... 4123
Delete ... 4125
DeleteRange ... 4126
At ... 4127
CompareArray ... 4128
InsertSort ... 4129
Search ... 4130
SearchGreat ... 4131
SearchLess ... 4132
SearchGreatOrEqual ... 4133
SearchLessOrEqual ... 4134
SearchFirst ... 4135
SearchLast ... 4136
Save ... 4137

© 2000-2025, MetaQuotes Ltd.

Load ... 4138
Type ... 4140
CList ... 4141
FreeMode ... 4143
FreeMode ... 4144
Total ... 4146
IsSorted ... 4147
SortMode ... 4148
CreateElement ... 4149
Add ... 4150
Insert ... 4151
DetachCurrent ... 4153
DeleteCurrent ... 4154
Delete ... 4155
Clear ... 4156
IndexOf ... 4157
GetNodeAtIndex ... 4158
GetFirstNode ... 4159
GetPrevNode ... 4160
GetCurrentNode ... 4161
GetNextNode ... 4162
GetLastNode ... 4163
Sort ... 4164
MoveToIndex ... 4165
Exchange ... 4166
CompareList ... 4167
Search ... 4168
Save ... 4169
Load ... 4171
Type ... 4173
CTreeNode ... 4174
Owner ... 4179
Left ... 4180
Right ... 4181
Balance ... 4182
BalanceL ... 4183
BalanceR ... 4184
CreateSample ... 4185
RefreshBalance ... 4186
GetNext ... 4187
SaveNode ... 4188
LoadNode ... 4189
Type ... 4190
CTree ... 4191
Root ... 4197
CreateElement ... 4198
Insert ... 4199
Detach ... 4200
Delete ... 4201
Clear ... 4202
Find ... 4203
Save ... 4204
Load ... 4205
Type ... 4206
Colecciones de datos genéricas ... 4207
ICollection<T> ... 4210
Add ... 4211
Count ... 4212

© 2000-2025, MetaQuotes Ltd.

Contains ... 4213
CopyTo ... 4214
Clear ... 4215
Remove ... 4216
IEqualityComparable<T> ... 4217
Equals ... 4218
HashCode ... 4219
IComparable<T> ... 4220
Compare ... 4221
IComparer<T> ... 4222
Compare ... 4223
IEqualityComparer<T> ... 4224
Equals ... 4225
HashCode ... 4226
IList<T> ... 4227
TryGetValue ... 4228
TrySetValue ... 4229
Insert ... 4230
IndexOf ... 4231
LastIndexOf ... 4232
RemoveAt ... 4233
IMap<TKey,TValue> ... 4234
Add ... 4235
Contains ... 4236
Remove ... 4237
TryGetValue ... 4238
TrySetValue ... 4239
CopyTo ... 4240
ISet<T> ... 4241
ExceptWith ... 4243
IntersectWith ... 4244
SymmetricExceptWith ... 4245
UnionWith ... 4246
IsProperSubsetOf ... 4247
IsProperSupersetOf ... 4248
IsSubsetOf ... 4249
IsSupersetOf ... 4250
Overlaps ... 4251
SetEquals ... 4252
CDefaultComparer<T> ... 4253
Compare ... 4254
CDefaultEqualityComparer<T> ... 4255
Equals ... 4256
HashCode ... 4257
CRedBlackTreeNode<T> ... 4258
Value ... 4259
Parent ... 4260
Left ... 4261
Right ... 4262
Color ... 4263
IsLeaf ... 4264
CreateEmptyNode ... 4265
CLinkedListNode<T> ... 4266
List ... 4267
Next ... 4268
Previous ... 4269
Value ... 4270
CKeyValuePair<TKey,TValue> ... 4271

© 2000-2025, MetaQuotes Ltd.

Key ... 4272
Value ... 4273
Clone ... 4274
Compare ... 4275
Equals ... 4276
HashCode ... 4277
CArrayList<T> ... 4278
Capacity ... 4280
Count ... 4281
Contains ... 4282
TrimExcess ... 4283
TryGetValue ... 4284
TrySetValue ... 4285
Add ... 4286
AddRange ... 4287
Insert ... 4288
InsertRange ... 4289
CopyTo ... 4290
BinarySearch ... 4291
IndexOf ... 4292
LastIndexOf ... 4293
Clear ... 4294
Remove ... 4295
RemoveAt ... 4296
RemoveRange ... 4297
Reverse ... 4298
Sort ... 4299
CHashMap<TKey,TValue> ... 4300
Add ... 4302
Count ... 4303
Comparer ... 4304
Contains ... 4305
ContainsKey ... 4306
ContainsValue ... 4307
CopyTo ... 4308
Clear ... 4309
Remove ... 4310
TryGetValue ... 4311
TrySetValue ... 4312
CHashSet<T> ... 4313
Add ... 4315
Count ... 4316
Contains ... 4317
Comparer ... 4318
TrimExcess ... 4319
CopyTo ... 4320
Clear ... 4321
Remove ... 4322
ExceptWith ... 4323
IntersectWith ... 4324
SymmetricExceptWith ... 4325
UnionWith ... 4326
IsProperSubsetOf ... 4327
IsProperSupersetOf ... 4328
IsSubsetOf ... 4329
IsSupersetOf ... 4330
Overlaps ... 4331
SetEquals ... 4332

© 2000-2025, MetaQuotes Ltd.

CLinkedList<T> 4333
Add 4335
AddAfter 4336
AddBefore 4337
AddFirst 4338
AddLast 4339
Count 4340
Head 4341
First 4342
Last 4343
Contains 4344
CopyTo 4345
Clear 4346
Remove 4347
RemoveFirst 4348
RemoveLast 4349
Find 4350
FindLast 4351
CQueue<T> 4352
Add 4353
Enqueue 4354
Count 4355
Contains 4356
TrimExcess 4357
CopyTo 4358
Clear 4359
Remove 4360
Dequeue 4361
Peek 4362
CRedBlackTree<T> 4363
Add 4365
Count 4366
Root 4367
Contains 4368
Comparer 4369
TryGetMin 4370
TryGetMax 4371
CopyTo 4372
Clear 4373
Remove 4374
RemoveMin 4375
RemoveMax 4376
Find 4377
FindMin 4378
FindMax 4379
CSortedMap<TKey, TValue> 4380
Add 4382
Count 4383
Comparer 4384
Contains 4385
ContainsKey 4386
ContainsValue 4387
CopyTo 4388
Clear 4389
Remove 4390
TryGetValue 4391
TrySetValue 4392
CSortedSet<T> 4393

© 2000-2025, MetaQuotes Ltd.

Add ... 4395
Count ... 4396
Contains ... 4397
Comparer ... 4398
TryGetMin ... 4399
TryGetMax ... 4400
CopyTo ... 4401
Clear ... 4402
Remove ... 4403
ExceptWith ... 4404
IntersectWith ... 4405
SymmetricExceptWith ... 4406
UnionWith ... 4407
IsProperSubsetOf ... 4408
IsProperSupersetOf ... 4409
IsSubsetOf ... 4410
IsSupersetOf ... 4411
Overlaps ... 4412
SetEquals ... 4413
GetViewBetween ... 4414
GetReverse ... 4415
CStack<T> ... 4416
Add ... 4417
Count ... 4418
Contains ... 4419
TrimExcess ... 4420
CopyTo ... 4421
Clear ... 4422
Remove ... 4423
Push ... 4424
Peek ... 4425
Pop ... 4426
ArrayBinarySearch<T> ... 4427
ArrayIndexOf<T> ... 4428
ArrayLastIndexOf<T> ... 4429
ArrayReverse<T> ... 4430
Compare ... 4431
Equals<T> ... 4434
GetHashCode ... 4435
Archivos ... 4438
CFile ... 4439
Handle ... 4441
Filename ... 4442
Flags ... 4443
SetUnicode ... 4444
SetCommon ... 4445
Open ... 4446
Close ... 4447
Delete ... 4448
IsExist ... 4449
Copy ... 4450
Move ... 4451
Size ... 4452
Tell ... 4453
Seek ... 4454
Flush ... 4455
IsEnding ... 4456
IsLineEnding ... 4457

© 2000-2025, MetaQuotes Ltd.

FolderCreate ... 4458
FolderDelete ... 4459
FolderClean ... 4460
FileFindFirst ... 4461
FileFindNext ... 4462
FileFindClose ... 4463
CFileBin ... 4464
Open ... 4466
WriteChar ... 4467
WriteShort ... 4468
WriteInteger ... 4469
WriteLong ... 4470
WriteFloat ... 4471
WriteDouble ... 4472
WriteString ... 4473
WriteCharArray ... 4474
WriteShortArray ... 4475
WriteIntegerArray ... 4476
WriteLongArray ... 4477
WriteFloatArray ... 4478
WriteDoubleArray ... 4479
WriteObject ... 4480
ReadChar ... 4481
ReadShort ... 4482
ReadInteger ... 4483
ReadLong ... 4484
ReadFloat ... 4485
ReadDouble ... 4486
ReadString ... 4487
ReadCharArray ... 4488
ReadShortArray ... 4489
ReadIntegerArray ... 4490
ReadLongArray ... 4491
ReadFloatArray ... 4492
ReadDoubleArray ... 4493
ReadObject ... 4494
CFileTxt ... 4495
Open ... 4496
WriteString ... 4497
ReadString ... 4498
Cadenas de caracteres ... 4499
CString ... 4500
Str ... 4502
Len ... 4503
Copy ... 4504
Fill ... 4505
Assign ... 4506
Append ... 4507
Insert ... 4508
Compare ... 4509
CompareNoCase ... 4510
Left ... 4511
Right ... 4512
Mid ... 4513
Trim ... 4514
TrimLeft ... 4515
TrimRight ... 4516
Clear ... 4517

© 2000-2025, MetaQuotes Ltd.

ToUpper ... 4518
ToLower ... 4519
Reverse ... 4520
Find ... 4521
FindRev ... 4522
Remove ... 4523
Replace ... 4524
Objetos gráficos ... 4525
CChartObject ... 4526
ChartId ... 4529
Window ... 4530
Name ... 4531
NumPoints ... 4532
Attach ... 4533
SetPoint ... 4534
Delete ... 4535
Detach ... 4536
ShiftObject ... 4537
ShiftPoint ... 4538
Time ... 4539
Price ... 4541
Color ... 4543
Style ... 4544
Width ... 4545
Background ... 4546
Selected ... 4547
Selectable ... 4548
Description ... 4549
Tooltip ... 4550
Timeframes ... 4551
Z_Order ... 4552
CreateTime ... 4553
LevelsCount ... 4554
LevelColor ... 4555
LevelStyle ... 4557
LevelWidth ... 4559
LevelValue ... 4561
LevelDescription ... 4563
GetInteger ... 4565
SetInteger ... 4567
GetDouble ... 4569
SetDouble ... 4571
GetString ... 4573
SetString ... 4575
Save ... 4577
Load ... 4578
Type ... 4579
Objects Lines ... 4580
CChartObjectVLine ... 4581
Create ... 4582
Type ... 4583
CChartObjectHLine ... 4584
Create ... 4585
Type ... 4586
CChartObjectTrend ... 4587
Create ... 4589
RayLeft ... 4590
RayRight ... 4591

© 2000-2025, MetaQuotes Ltd.

Save ... 4592
Load ... 4593
Type ... 4594
CChartObjectTrendByAngle ... 4595
Create ... 4597
Angle ... 4598
Type ... 4599
CChartObjectCycles ... 4600
Create ... 4601
Type ... 4602
Objects Channels ... 4603
CChartObjectChannel ... 4604
Create ... 4606
Type ... 4607
CChartObjectRegression ... 4608
Create ... 4610
Type ... 4611
CChartObjectStdDevChannel ... 4612
Create ... 4614
Deviations ... 4615
Save ... 4616
Load ... 4617
Type ... 4618
CChartObjectPitchfork ... 4619
Create ... 4620
Type ... 4621
Gann Tools ... 4622
CChartObjectGannLine ... 4623
Create ... 4625
PipsPerBar ... 4626
Save ... 4627
Load ... 4628
Type ... 4629
CChartObjectGannFan ... 4630
Create ... 4632
PipsPerBar ... 4633
Downtrend ... 4634
Save ... 4635
Load ... 4636
Type ... 4637
CChartObjectGannGrid ... 4638
Create ... 4640
PipsPerBar ... 4641
Downtrend ... 4642
Save ... 4643
Load ... 4644
Type ... 4645
Fibonacci Tools ... 4646
CChartObjectFibo ... 4647
Create ... 4649
Type ... 4650
CChartObjectFiboTimes ... 4651
Create ... 4652
Type ... 4653
CChartObjectFiboFan ... 4654
Create ... 4655
Type ... 4656
CChartObjectFiboArc ... 4657

© 2000-2025, MetaQuotes Ltd.

Create ... 4659
Scale ... 4660
Ellipse ... 4661
Save ... 4662
Load ... 4663
Type ... 4664
CChartObjectFiboChannel ... 4665
Create ... 4667
Type ... 4668
CChartObjectFiboExpansion ... 4669
Create ... 4671
Type ... 4672
Elliott Tools ... 4673
CChartObjectElliottWave3 ... 4674
Create ... 4676
Degree ... 4677
Lines ... 4678
Save ... 4679
Load ... 4680
Type ... 4681
CChartObjectElliottWave5 ... 4682
Create ... 4684
Type ... 4686
Objects Shapes ... 4687
CChartObjectRectangle ... 4688
Create ... 4689
Type ... 4690
CChartObjectTriangle ... 4691
Create ... 4692
Type ... 4693
CChartObjectEllipse ... 4694
Create ... 4695
Type ... 4696
Objects Arrows ... 4697
CChartObjectArrow ... 4698
Create ... 4700
ArrowCode ... 4702
Anchor ... 4704
Save ... 4706
Load ... 4707
Type ... 4708
Arrows with fixed code ... 4709
Create ... 4711
ArrowCode ... 4713
Type ... 4714
Objects Controls ... 4715
CChartObjectText ... 4716
Create ... 4718
Angle ... 4719
Font ... 4720
FontSize ... 4721
Anchor ... 4722
Save ... 4723
Load ... 4724
Type ... 4725
CChartObjectLabel ... 4726
Create ... 4728
X_Distance ... 4729

© 2000-2025, MetaQuotes Ltd.

Y_Distance ... 4730
X_Size ... 4731
Y_Size ... 4732
Corner ... 4733
Time ... 4734
Price ... 4735
Save ... 4736
Load ... 4737
Type ... 4738
CChartObjectEdit ... 4739
Create ... 4741
TextAlign ... 4742
X_Size ... 4743
Y_Size ... 4744
BackColor ... 4745
BorderColor ... 4746
ReadOnly ... 4747
Angle ... 4748
Save ... 4749
Load ... 4750
Type ... 4751
CChartObjectButton ... 4752
State ... 4754
Save ... 4755
Load ... 4756
Type ... 4757
CChartObjectSubChart ... 4758
Create ... 4760
X_Distance ... 4761
Y_Distance ... 4762
Corner ... 4763
X_Size ... 4764
Y_Size ... 4765
Symbol ... 4766
Period ... 4767
Scale ... 4768
DateScale ... 4769
PriceScale ... 4770
Time ... 4771
Price ... 4772
Save ... 4773
Load ... 4774
Type ... 4775
CChartObjectBitmap ... 4776
Create ... 4778
BmpFile ... 4779
X_Offset ... 4780
Y_Offset ... 4781
Save ... 4782
Load ... 4783
Type ... 4784
CChartObjectBmpLabel ... 4785
Create ... 4787
X_Distance ... 4788
Y_Distance ... 4789
X_Offset ... 4790
Y_Offset ... 4791
Corner ... 4792

© 2000-2025, MetaQuotes Ltd.

X_Size ... 4793
Y_Size ... 4794
BmpFileOn ... 4795
BmpFileOff ... 4796
State ... 4797
Time ... 4798
Price ... 4799
Save ... 4800
Load ... 4801
Type ... 4802
CChartObjectRectLabel ... 4803
Create ... 4805
X_Size ... 4806
Y_Size ... 4807
BackColor ... 4808
Angle ... 4809
BorderType ... 4810
Save ... 4811
Load ... 4812
Type ... 4813
Gráficos personalizados ... 4814
CCanvas ... 4815
Attach ... 4819
Arc ... 4820
Pie ... 4824
FillPolygon ... 4828
FillEllipse ... 4829
GetDefaultColor ... 4830
ChartObjectName ... 4831
Circle ... 4832
CircleAA ... 4833
CircleWu ... 4834
Create ... 4835
CreateBitmap ... 4836
CreateBitmapLabel ... 4838
Destroy ... 4840
Ellipse ... 4841
EllipseAA ... 4842
EllipseWu ... 4843
Erase ... 4844
Fill ... 4845
FillCircle ... 4846
FillRectangle ... 4847
FillTriangle ... 4848
FontAngleGet ... 4849
FontAngleSet ... 4850
FontFlagsGet ... 4851
FontFlagsSet ... 4852
FontGet ... 4853
FontNameGet ... 4854
FontNameSet ... 4855
FontSet ... 4856
FontSizeGet ... 4857
FontSizeSet ... 4858
Height ... 4859
Line ... 4860
LineAA ... 4861
LineWu ... 4862

© 2000-2025, MetaQuotes Ltd.

LineHorizontal ... 4863
LineVertical ... 4864
LineStyleSet ... 4865
LineThick ... 4866
LineThickVertical ... 4868
LineThickHorizontal ... 4869
LoadFromFile ... 4870
PixelGet ... 4871
PixelSet ... 4872
PixelSetAA ... 4873
Polygon ... 4874
PolygonAA ... 4875
PolygonWu ... 4876
PolygonThick ... 4877
PolygonSmooth ... 4878
Polyline ... 4879
PolylineSmooth ... 4880
PolylineThick ... 4881
PolylineWu ... 4882
PolylineAA ... 4883
Rectangle ... 4884
Resize ... 4885
ResourceName ... 4886
TextHeight ... 4887
TextOut ... 4888
TextSize ... 4889
TextWidth ... 4890
TransparentLevelSet ... 4891
Triangle ... 4892
TriangleAA ... 4893
TriangleWu ... 4894
Update ... 4895
Width ... 4896
CChartCanvas ... 4897
ColorBackground ... 4901
ColorBorder ... 4902
ColorText ... 4903
ColorGrid ... 4904
MaxData ... 4905
MaxDescrLen ... 4906
ShowFlags ... 4907
IsShowLegend ... 4908
IsShowScaleLeft ... 4909
IsShowScaleRight ... 4910
IsShowScaleTop ... 4911
IsShowScaleBottom ... 4912
IsShowGrid ... 4913
IsShowDescriptors ... 4914
IsShowPercent ... 4915
VScaleMin ... 4916
VScaleMax ... 4917
NumGrid ... 4918
DataOffset ... 4919
DataTotal ... 4920
DrawDescriptors ... 4921
DrawData ... 4922
Create ... 4923
AllowedShowFlags ... 4924

© 2000-2025, MetaQuotes Ltd.

ShowLegend 4925
ShowScaleLeft 4926
ShowScaleRight 4927
ShowScaleTop 4928
ShowScaleBottom 4929
ShowGrid 4930
ShowDescriptors 4931
ShowValue 4932
ShowPercent 4933
LegendAlignment 4934
Accumulative 4935
VScaleParams 4936
DescriptorUpdate 4937
ColorUpdate 4938
ValuesCheck 4939
Redraw 4940
DrawBackground 4941
DrawLegend 4942
DrawLegendVertical 4943
DrawLegendHorizontal 4944
CalcScales 4945
DrawScales 4946
DrawScaleLeft 4947
DrawScaleRight 4948
DrawScaleTop 4949
DrawScaleBottom 4950
DrawGrid 4951
DrawChart 4952
CHistogramChart 4953
Gradient 4958
BarGap 4959
BarMinSize 4960
BarBorder 4961
Create 4962
SeriesAdd 4963
SeriesInsert 4964
SeriesUpdate 4965
SeriesDelete 4966
ValueUpdate 4967
DrawData 4968
DrawBar 4969
GradientBrush 4970
CLineChart 4971
Filled 4975
Create 4976
SeriesAdd 4977
SeriesInsert 4978
SeriesUpdate 4979
SeriesDelete 4980
ValueUpdate 4981
DrawChart 4982
DrawData 4983
CalcArea 4984
CPieChart 4985
Create 4989
SeriesSet 4990
ValueAdd 4991
ValueInsert 4992

© 2000-2025, MetaQuotes Ltd.

ValueUpdate 4993
ValueDelete 4994
DrawChart 4995
DrawPie 4996
LabelMake 4997
Gráficos 3D 4998
CCanvas3D 4999
AmbientColorGet 5001
AmbientColorSet 5002
Attach 5003
Create 5004
Destroy 5005
DXContext 5006
DXDispatcher 5007
InputScene 5008
LightColorGet 5009
LightColorSet 5010
LightDirectionGet 5011
LightDirectionSet 5012
ObjectAdd 5013
ProjectionMatrixGet 5014
ProjectionMatrixSet 5015
Render 5016
RenderBegin 5017
RenderEnd 5018
ViewMatrixGet 5019
ViewMatrixSet 5020
ViewPositionSet 5021
ViewRotationSet 5022
ViewTargetSet 5023
ViewUpDirectionSet 5024
Gráficos de precios 5025
ChartID 5030
Mode 5031
Foreground 5032
Shift 5033
ShiftSize 5034
AutoScroll 5035
Scale 5036
ScaleFix 5037
ScaleFix_11 5038
FixedMax 5039
FixedMin 5040
PointsPerBar 5041
ScalePPB 5042
ShowOHLC 5043
ShowLineBid 5044
ShowLineAsk 5045
ShowLastLine 5046
ShowPeriodSep 5047
ShowGrid 5048
ShowVolumes 5049
ShowObjectDescr 5050
ShowDateScale 5051
ShowPriceScale 5052
ColorBackground 5053
ColorForeground 5054
ColorGrid 5055

© 2000-2025, MetaQuotes Ltd.

ColorBarUp ... 5056
ColorBarDown ... 5057
ColorCandleBull ... 5058
ColorCandleBear ... 5059
ColorChartLine ... 5060
ColorVolumes ... 5061
ColorLineBid ... 5062
ColorLineAsk ... 5063
ColorLineLast ... 5064
ColorStopLevels ... 5065
VisibleBars ... 5066
WindowsTotal ... 5067
WindowIsVisible ... 5068
WindowHandle ... 5069
FirstVisibleBar ... 5070
WidthInBars ... 5071
WidthInPixels ... 5072
HeightInPixels ... 5073
PriceMin ... 5074
PriceMax ... 5075
Attach ... 5076
FirstChart ... 5077
NextChart ... 5078
Open ... 5079
Detach ... 5080
Close ... 5081
BringToTop ... 5082
EventObjectCreate ... 5083
EventObjectDelete ... 5084
IndicatorAdd ... 5085
IndicatorDelete ... 5086
IndicatorsTotal ... 5087
IndicatorName ... 5088
Navigate ... 5089
Symbol ... 5090
Period ... 5091
Redraw ... 5092
GetInteger ... 5093
SetInteger ... 5094
GetDouble ... 5095
SetDouble ... 5096
GetString ... 5097
SetString ... 5098
SetSymbolPeriod ... 5099
ApplyTemplate ... 5100
ScreenShot ... 5101
WindowOnDropped ... 5102
PriceOnDropped ... 5103
TimeOnDropped ... 5104
XOnDropped ... 5105
YOnDropped ... 5106
Save ... 5107
Load ... 5108
Type ... 5109
Gráficos científicos ... 5110
GraphPlot ... 5111
CAxis ... 5115
AutoScale ... 5117

© 2000-2025, MetaQuotes Ltd.

Min ... 5118
Max ... 5119
Step ... 5120
Name ... 5121
Color ... 5122
ValuesSize ... 5123
ValuesWidth ... 5124
ValuesFormat ... 5125
ValuesDateTimeMode ... 5126
ValuesFunctionFormat ... 5127
ValuesFunctionFormatCBData ... 5129
NameSize ... 5130
ZeroLever ... 5131
DefaultStep ... 5132
MaxLabels ... 5133
MinGrace ... 5134
MaxGrace ... 5135
SelectAxisScale ... 5136
CColorGenerator ... 5137
Next ... 5138
Reset ... 5139
CCurve ... 5140
Type ... 5142
Name ... 5143
Color ... 5144
XMax ... 5145
XMin ... 5146
YMax ... 5147
YMin ... 5148
Size ... 5149
PointsSize ... 5150
PointsFill ... 5151
PointsColor ... 5152
GetX ... 5153
GetY ... 5154
LinesStyle ... 5155
LinesIsSmooth ... 5156
LinesSmoothTension ... 5157
LinesSmoothStep ... 5158
LinesEndStyle ... 5159
LinesWidth ... 5160
HistogramWidth ... 5162
CustomPlotCBData ... 5163
CustomPlotFunction ... 5164
PointsType ... 5168
StepsDimension ... 5169
TrendLineCoefficients ... 5170
TrendLineColor ... 5171
TrendLineVisible ... 5172
Update ... 5174
Visible ... 5176
CGraphic ... 5177
Create ... 5180
Destroy ... 5181
Update ... 5182
ChartObjectName ... 5183
ResourceName ... 5184
XAxis ... 5185

© 2000-2025, MetaQuotes Ltd.

YAxis ... 5186
GapSize ... 5187
BackgroundColor ... 5188
BackgroundMain ... 5189
BackgroundMainSize ... 5190
BackgroundMainColor ... 5191
BackgroundSub ... 5192
BackgroundSubSize ... 5193
BackgroundSubColor ... 5194
GridLineColor ... 5195
GridBackgroundColor ... 5196
GridCircleRadius ... 5197
GridCircleColor ... 5198
GridHasCircle ... 5199
GridAxisLineColor ... 5200
HistoryNameWidth ... 5201
HistoryNameSize ... 5202
HistorySymbolSize ... 5203
TextAdd ... 5204
LineAdd ... 5205
CurveAdd ... 5206
CurvePlot ... 5209
CurvePlotAll ... 5210
CurveGetByIndex ... 5211
CurveGetByName ... 5212
CurveRemoveByName ... 5213
CurveRemoveByIndex ... 5214
CurvesTotal ... 5215
MarksToAxisAdd ... 5216
MajorMarkSize ... 5217
FontSet ... 5218
FontGet ... 5219
Attach ... 5220
CalculateMaxMinValues ... 5221
Height ... 5222
IndentDown ... 5223
IndentLeft ... 5224
IndentRight ... 5225
IndentUp ... 5226
Redraw ... 5227
ResetParameters ... 5228
ScaleX ... 5229
ScaleY ... 5230
SetDefaultParameters ... 5231
Width ... 5232
Indicadores ... 5233
Clases base ... 5234
CSpreadBuffer ... 5235
Size ... 5237
SetSymbolPeriod ... 5238
At ... 5239
Refresh ... 5240
RefreshCurrent ... 5241
CTimeBuffer ... 5242
Size ... 5244
SetSymbolPeriod ... 5245
At ... 5246
Refresh ... 5247

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent ... 5248
CTickVolumeBuffer ... 5249
Size ... 5251
SetSymbolPeriod ... 5252
At ... 5253
Refresh ... 5254
RefreshCurrent ... 5255
CRealVolumeBuffer ... 5256
Size ... 5258
SetSymbolPeriod ... 5259
At ... 5260
Refresh ... 5261
RefreshCurrent ... 5262
CDoubleBuffer ... 5263
Size ... 5265
SetSymbolPeriod ... 5266
At ... 5267
Refresh ... 5268
RefreshCurrent ... 5269
COpenBuffer ... 5270
Refresh ... 5271
RefreshCurrent ... 5272
CHighBuffer ... 5273
Refresh ... 5274
RefreshCurrent ... 5275
CLowBuffer ... 5276
Refresh ... 5277
RefreshCurrent ... 5278
CCloseBuffer ... 5279
Refresh ... 5280
RefreshCurrent ... 5281
CIndicatorBuffer ... 5282
Offset ... 5284
Name ... 5285
At ... 5286
Refresh ... 5287
RefreshCurrent ... 5288
CSeries ... 5289
Name ... 5291
BuffersTotal ... 5292
Timeframe ... 5293
Symbol ... 5294
Period ... 5295
RefreshCurrent ... 5296
BufferSize ... 5297
BufferResize ... 5298
Refresh ... 5299
PeriodDescription ... 5300
CPriceSeries ... 5301
BufferResize ... 5303
GetData ... 5304
Refresh ... 5305
MinIndex ... 5306
MinValue ... 5307
MaxIndex ... 5308
MaxValue ... 5309
CIndicator ... 5310
Handle ... 5313

© 2000-2025, MetaQuotes Ltd.

Status ... 5314
FullRelease ... 5315
Create ... 5316
BufferResize ... 5317
BarsCalculated ... 5318
GetData ... 5319
Refresh ... 5322
Minimum ... 5323
MinValue ... 5324
Maximum ... 5325
MaxValue ... 5326
MethodDescription ... 5327
PriceDescription ... 5328
VolumeDescription ... 5329
AddToChart ... 5330
DeleteFromChart ... 5331
CIndicators ... 5332
Create ... 5333
Refresh ... 5334
Clases de series temporales ... 5335
CiSpread ... 5336
Create ... 5338
BufferResize ... 5339
GetData ... 5340
Refresh ... 5342
CiTime ... 5343
Create ... 5345
BufferResize ... 5346
GetData ... 5347
Refresh ... 5349
CiTickVolume ... 5350
Create ... 5352
BufferResize ... 5353
GetData ... 5354
Refresh ... 5356
CiRealVolume ... 5357
Create ... 5359
BufferResize ... 5360
GetData ... 5361
Refresh ... 5363
CiOpen ... 5364
Create ... 5366
GetData ... 5367
CiHigh ... 5369
Create ... 5371
GetData ... 5372
CiLow ... 5374
Create ... 5376
GetData ... 5377
CiClose ... 5379
Create ... 5381
GetData ... 5382
Indicadores de tendencia ... 5384
CiADX ... 5385
MaPeriod ... 5387
Create ... 5388
Main ... 5389
Plus ... 5390

© 2000-2025, MetaQuotes Ltd.

Minus ... 5391
Type ... 5392
CiADXWilder ... 5393
MaPeriod ... 5395
Create ... 5396
Main ... 5397
Plus ... 5398
Minus ... 5399
Type ... 5400
CiBands ... 5401
MaPeriod ... 5403
MaShift ... 5404
Deviation ... 5405
Applied ... 5406
Create ... 5407
Base ... 5408
Upper ... 5409
Lower ... 5410
Type ... 5411
CiEnvelopes ... 5412
MaPeriod ... 5414
MaShift ... 5415
MaMethod ... 5416
Deviation ... 5417
Applied ... 5418
Create ... 5419
Upper ... 5420
Lower ... 5421
Type ... 5422
CiIchimoku ... 5423
TenkanSenPeriod ... 5425
KijunSenPeriod ... 5426
SenkouSpanBPeriod ... 5427
Create ... 5428
TenkanSen ... 5429
KijunSen ... 5430
SenkouSpanA ... 5431
SenkouSpanB ... 5432
ChinkouSpan ... 5433
Type ... 5434
CiMA ... 5435
MaPeriod ... 5437
MaShift ... 5438
MaMethod ... 5439
Applied ... 5440
Create ... 5441
Main ... 5442
Type ... 5443
CiSAR ... 5444
SarStep ... 5446
Maximum ... 5447
Create ... 5448
Main ... 5449
Type ... 5450
CiStdDev ... 5451
MaPeriod ... 5453
MaShift ... 5454
MaMethod ... 5455

© 2000-2025, MetaQuotes Ltd.

Applied ... 5456
Create ... 5457
Main ... 5458
Type ... 5459
CiDEMA ... 5460
MaPeriod ... 5462
IndShift ... 5463
Applied ... 5464
Create ... 5465
Main ... 5466
Type ... 5467
CiTEMA ... 5468
MaPeriod ... 5470
IndShift ... 5471
Applied ... 5472
Create ... 5473
Main ... 5474
Type ... 5475
CiFrAMA ... 5476
MaPeriod ... 5478
IndShift ... 5479
Applied ... 5480
Create ... 5481
Main ... 5482
Type ... 5483
CiAMA ... 5484
MaPeriod ... 5486
FastEmaPeriod ... 5487
SlowEmaPeriod ... 5488
IndShift ... 5489
Applied ... 5490
Create ... 5491
Main ... 5492
Type ... 5493
CiVIDyA ... 5494
CmoPeriod ... 5496
EmaPeriod ... 5497
IndShift ... 5498
Applied ... 5499
Create ... 5500
Main ... 5501
Type ... 5502
Osciladores ... 5503
CiATR ... 5504
MaPeriod ... 5506
Create ... 5507
Main ... 5508
Type ... 5509
CiBearsPower ... 5510
MaPeriod ... 5512
Create ... 5513
Main ... 5514
Type ... 5515
CiBullsPower ... 5516
MaPeriod ... 5518
Create ... 5519
Main ... 5520
Type ... 5521

© 2000-2025, MetaQuotes Ltd.

CiCCI ... 5522
MaPeriod ... 5524
Applied ... 5525
Create ... 5526
Main ... 5527
Type ... 5528
CiChaikin ... 5529
FastMaPeriod ... 5531
SlowMaPeriod ... 5532
MaMethod ... 5533
Applied ... 5534
Create ... 5535
Main ... 5536
Type ... 5537
CiDeMarker ... 5538
MaPeriod ... 5540
Create ... 5541
Main ... 5542
Type ... 5543
CiForce ... 5544
MaPeriod ... 5546
MaMethod ... 5547
Applied ... 5548
Create ... 5549
Main ... 5550
Type ... 5551
CiMACD ... 5552
FastEmaPeriod ... 5554
SlowEmaPeriod ... 5555
SignalPeriod ... 5556
Applied ... 5557
Create ... 5558
Main ... 5559
Signal ... 5560
Type ... 5561
CiMomentum ... 5562
MaPeriod ... 5564
Applied ... 5565
Create ... 5566
Main ... 5567
Type ... 5568
CiOsMA ... 5569
FastEmaPeriod ... 5571
SlowEmaPeriod ... 5572
SignalPeriod ... 5573
Applied ... 5574
Create ... 5575
Main ... 5576
Type ... 5577
CiRSI ... 5578
MaPeriod ... 5580
Applied ... 5581
Create ... 5582
Main ... 5583
Type ... 5584
CiRVI ... 5585
MaPeriod ... 5587
Create ... 5588

© 2000-2025, MetaQuotes Ltd.

Main ... 5589
Signal ... 5590
Type ... 5591
CiStochastic ... 5592
Kperiod ... 5594
Dperiod ... 5595
Slowing ... 5596
MaMethod ... 5597
PriceField ... 5598
Create ... 5599
Main ... 5600
Signal ... 5601
Type ... 5602
CiTriX ... 5603
MaPeriod ... 5605
Applied ... 5606
Create ... 5607
Main ... 5608
Type ... 5609
CiWPR ... 5610
CalcPeriod ... 5612
Create ... 5613
Main ... 5614
Type ... 5615
Indicadores de volumen ... 5616
CiAD ... 5617
Applied ... 5619
Create ... 5620
Main ... 5621
Type ... 5622
CiMFI ... 5623
MaPeriod ... 5625
Applied ... 5626
Create ... 5627
Main ... 5628
Type ... 5629
CiOBV ... 5630
Applied ... 5632
Create ... 5633
Main ... 5634
Type ... 5635
CiVolumes ... 5636
Applied ... 5638
Create ... 5639
Main ... 5640
Type ... 5641
Indicadores de Bill Williams ... 5642
CiAC ... 5643
Create ... 5645
Main ... 5646
Type ... 5647
CiAlligator ... 5648
JawPeriod ... 5650
JawShift ... 5651
TeethPeriod ... 5652
TeethShift ... 5653
LipsPeriod ... 5654
LipsShift ... 5655

© 2000-2025, MetaQuotes Ltd.

MaMethod ... 5656
Applied ... 5657
Create ... 5658
Jaw ... 5659
Teeth ... 5660
Lips ... 5661
Type ... 5662
CiAO ... 5663
Create ... 5665
Main ... 5666
Type ... 5667
CiFractals ... 5668
Create ... 5670
Upper ... 5671
Lower ... 5672
Type ... 5673
CiGator ... 5674
JawPeriod ... 5676
JawShift ... 5677
TeethPeriod ... 5678
TeethShift ... 5679
LipsPeriod ... 5680
LipsShift ... 5681
MaMethod ... 5682
Applied ... 5683
Create ... 5684
Upper ... 5685
Lower ... 5686
Type ... 5687
CiBWMFI ... 5688
Applied ... 5690
Create ... 5691
Main ... 5692
Type ... 5693
Indicadores personalizados ... 5694
NumBuffers ... 5695
NumParams ... 5696
ParamType ... 5697
ParamLong ... 5698
ParamDouble ... 5699
ParamString ... 5700
Type ... 5701
Clases de comercio ... 5702
CAccountInfo ... 5703
Login ... 5705
TradeMode ... 5706
TradeModeDescription ... 5707
Leverage ... 5708
StopoutMode ... 5709
StopoutModeDescription ... 5710
MarginMode ... 5711
MarginModeDescription ... 5712
TradeAllowed ... 5713
TradeExpert ... 5714
LimitOrders ... 5715
Balance ... 5716
Credit ... 5717
Profit ... 5718

© 2000-2025, MetaQuotes Ltd.

Equity ... 5719
Margin ... 5720
FreeMargin ... 5721
MarginLevel ... 5722
MarginCall ... 5723
MarginStopOut ... 5724
Name ... 5725
Server ... 5726
Currency ... 5727
Company ... 5728
InfoInteger ... 5729
InfoDouble ... 5730
InfoString ... 5731
OrderProfitCheck ... 5732
MarginCheck ... 5733
FreeMarginCheck ... 5734
MaxLotCheck ... 5735
CSymbolInfo ... 5736
Refresh ... 5740
RefreshRates ... 5741
Name ... 5742
Select ... 5743
IsSynchronized ... 5744
Volume ... 5745
VolumeHigh ... 5746
VolumeLow ... 5747
Time ... 5748
Spread ... 5749
SpreadFloat ... 5750
TicksBookDepth ... 5751
StopsLevel ... 5752
FreezeLevel ... 5753
Bid ... 5754
BidHigh ... 5755
BidLow ... 5756
Ask ... 5757
AskHigh ... 5758
AskLow ... 5759
Last ... 5760
LastHigh ... 5761
LastLow ... 5762
TradeCalcMode ... 5763
TradeCalcModeDescription ... 5764
TradeMode ... 5765
TradeModeDescription ... 5766
TradeExecution ... 5767
TradeExecutionDescription ... 5768
SwapMode ... 5769
SwapModeDescription ... 5770
SwapRollover3days ... 5771
SwapRollover3daysDescription ... 5772
MarginInitial ... 5773
MarginMaintenance ... 5774
MarginLong ... 5775
MarginShort ... 5776
MarginLimit ... 5777
MarginStop ... 5778
MarginStopLimit ... 5779

© 2000-2025, MetaQuotes Ltd.

TradeTimeFlags ... 5780
TradeFillFlags ... 5781
Digits ... 5782
Point ... 5783
TickValue ... 5784
TickValueProfit ... 5785
TickValueLoss ... 5786
TickSize ... 5787
ContractSize ... 5788
LotsMin ... 5789
LotsMax ... 5790
LotsStep ... 5791
LotsLimit ... 5792
SwapLong ... 5793
SwapShort ... 5794
CurrencyBase ... 5795
CurrencyProfit ... 5796
CurrencyMargin ... 5797
Bank ... 5798
Description ... 5799
Path ... 5800
SessionDeals ... 5801
SessionBuyOrders ... 5802
SessionSellOrders ... 5803
SessionTurnover ... 5804
SessionInterest ... 5805
SessionBuyOrdersVolume ... 5806
SessionSellOrdersVolume ... 5807
SessionOpen ... 5808
SessionClose ... 5809
SessionAW ... 5810
SessionPriceSettlement ... 5811
SessionPriceLimitMin ... 5812
SessionPriceLimitMax ... 5813
InfoInteger ... 5814
InfoDouble ... 5815
InfoString ... 5816
NormalizePrice ... 5817
COrderInfo ... 5818
Ticket ... 5820
TimeSetup ... 5821
TimeSetupMsc ... 5822
OrderType ... 5823
TypeDescription ... 5824
State ... 5825
StateDescription ... 5826
TimeExpiration ... 5827
TimeDone ... 5828
TimeDoneMsc ... 5829
TypeFilling ... 5830
TypeFillingDescription ... 5831
TypeTime ... 5832
TypeTimeDescription ... 5833
Magic ... 5834
PositionId ... 5835
VolumeInitial ... 5836
VolumeCurrent ... 5837
PriceOpen ... 5838

© 2000-2025, MetaQuotes Ltd.

StopLoss ... 5839
TakeProfit ... 5840
PriceCurrent ... 5841
PriceStopLimit ... 5842
Symbol ... 5843
Comment ... 5844
InfoInteger ... 5845
InfoDouble ... 5846
InfoString ... 5847
StoreState ... 5848
CheckState ... 5849
Select ... 5850
SelectByIndex ... 5851
CHistoryOrderInfo ... 5852
TimeSetup ... 5854
TimeSetupMsc ... 5855
OrderType ... 5856
TypeDescription ... 5857
State ... 5858
StateDescription ... 5859
TimeExpiration ... 5860
TimeDone ... 5861
TimeDoneMsc ... 5862
TypeFilling ... 5863
TypeFillingDescription ... 5864
TypeTime ... 5865
TypeTimeDescription ... 5866
Magic ... 5867
PositionId ... 5868
VolumeInitial ... 5869
VolumeCurrent ... 5870
PriceOpen ... 5871
StopLoss ... 5872
TakeProfit ... 5873
PriceCurrent ... 5874
PriceStopLimit ... 5875
Symbol ... 5876
Comment ... 5877
InfoInteger ... 5878
InfoDouble ... 5879
InfoString ... 5880
Ticket ... 5881
SelectByIndex ... 5882
CPositionInfo ... 5883
Time ... 5885
TimeMsc ... 5886
TimeUpdate ... 5887
TimeUpdateMsc ... 5888
PositionType ... 5889
TypeDescription ... 5890
Magic ... 5891
Identifier ... 5892
Volume ... 5893
PriceOpen ... 5894
StopLoss ... 5895
TakeProfit ... 5896
PriceCurrent ... 5897
Commission ... 5898

© 2000-2025, MetaQuotes Ltd.

Swap ... 5899
Profit ... 5900
Symbol ... 5901
Comment ... 5902
InfoInteger ... 5903
InfoDouble ... 5904
InfoString ... 5905
Select ... 5906
SelectByIndex ... 5907
SelectByMagic ... 5908
SelectByTicket ... 5909
StoreState ... 5910
CheckState ... 5911
CDealInfo ... 5912
Order ... 5914
Time ... 5915
TimeMsc ... 5916
DealType ... 5917
TypeDescription ... 5918
Entry ... 5919
EntryDescription ... 5920
Magic ... 5921
PositionId ... 5922
Volume ... 5923
Price ... 5924
Commision ... 5925
Swap ... 5926
Profit ... 5927
Symbol ... 5928
Comment ... 5929
InfoInteger ... 5930
InfoDouble ... 5931
InfoString ... 5932
Ticket ... 5933
SelectByIndex ... 5934
CTrade ... 5935
LogLevel ... 5939
SetExpertMagicNumber ... 5940
SetDeviationInPoints ... 5941
SetTypeFilling ... 5942
SetTypeFillingBySymbol ... 5943
SetAsyncMode ... 5944
SetMarginMode ... 5945
OrderOpen ... 5946
OrderModify ... 5948
OrderDelete ... 5949
PositionOpen ... 5950
PositionModify ... 5951
PositionClose ... 5953
PositionClosePartial ... 5954
PositionCloseBy ... 5956
Buy ... 5957
Sell ... 5958
BuyLimit ... 5959
BuyStop ... 5961
SellLimit ... 5962
SellStop ... 5964
Request ... 5965

© 2000-2025, MetaQuotes Ltd.

RequestAction ... 5966
RequestActionDescription ... 5967
RequestMagic ... 5968
RequestOrder ... 5969
RequestSymbol ... 5970
RequestVolume ... 5971
RequestPrice ... 5972
RequestStopLimit ... 5973
RequestSL ... 5974
RequestTP ... 5975
RequestDeviation ... 5976
RequestType ... 5977
RequestTypeDescription ... 5978
RequestTypeFilling ... 5979
RequestTypeFillingDescription ... 5980
RequestTypeTime ... 5981
RequestTypeTimeDescription ... 5982
RequestExpiration ... 5983
RequestComment ... 5984
RequestPosition ... 5985
RequestPositionBy ... 5986
Result ... 5987
ResultRetcode ... 5988
ResultRetcodeDescription ... 5989
ResultDeal ... 5990
ResultOrder ... 5991
ResultVolume ... 5992
ResultPrice ... 5993
ResultBid ... 5994
ResultAsk ... 5995
ResultComment ... 5996
CheckResult ... 5997
CheckResultRetcode ... 5998
CheckResultRetcodeDescription ... 5999
CheckResultBalance ... 6000
CheckResultEquity ... 6001
CheckResultProfit ... 6002
CheckResultMargin ... 6003
CheckResultMarginFree ... 6004
CheckResultMarginLevel ... 6005
CheckResultComment ... 6006
PrintRequest ... 6007
PrintResult ... 6008
FormatRequest ... 6009
FormatRequestResult ... 6010
CTerminalInfo ... 6011
Build ... 6013
IsConnected ... 6014
IsDLLsAllowed ... 6015
IsTradeAllowed ... 6016
IsEmailEnabled ... 6017
IsFtpEnabled ... 6018
MaxBars ... 6019
CodePage ... 6020
CPUCores ... 6021
MemoryPhysical ... 6022
MemoryTotal ... 6023
MemoryAvailable ... 6024

© 2000-2025, MetaQuotes Ltd.

MemoryUsed ... 6025
IsX64 ... 6026
OpenCLSupport ... 6027
DiskSpace ... 6028
Language ... 6029
Name ... 6030
Company ... 6031
Path ... 6032
DataPath ... 6033
CommonDataPath ... 6034
InfoInteger ... 6035
InfoString ... 6036
Módulos de estrategias ... 6037
Clases base de los Asesores Expertos ... 6040
CExpertBase ... 6041
InitPhase ... 6043
TrendType ... 6044
UsedSeries ... 6045
EveryTick ... 6046
Open ... 6047
High ... 6048
Low ... 6049
Close ... 6050
Spread ... 6051
Time ... 6052
TickVolume ... 6053
RealVolume ... 6054
Init ... 6055
Symbol ... 6056
Period ... 6057
Magic ... 6058
ValidationSettings ... 6059
SetPriceSeries ... 6060
SetOtherSeries ... 6061
InitIndicators ... 6062
InitOpen ... 6063
InitHigh ... 6064
InitLow ... 6065
InitClose ... 6066
InitSpread ... 6067
InitTime ... 6068
InitTickVolume ... 6069
InitRealVolume ... 6070
PriceLevelUnit ... 6071
StartIndex ... 6072
CompareMagic ... 6073
CExpert ... 6074
Init ... 6079
Magic ... 6080
InitSignal ... 6081
InitTrailing ... 6082
InitMoney ... 6083
InitTrade ... 6084
Deinit ... 6085
OnTickProcess ... 6086
OnTradeProcess ... 6087
OnTimerProcess ... 6088
OnChartEventProcess ... 6089

© 2000-2025, MetaQuotes Ltd.

OnBookEventProcess ... 6090
MaxOrders ... 6091
Signal ... 6092
ValidationSettings ... 6093
InitIndicators ... 6094
OnTick ... 6095
OnTrade ... 6096
OnTimer ... 6097
OnChartEvent ... 6098
OnBookEvent ... 6099
InitParameters ... 6100
DeinitTrade ... 6101
DeinitSignal ... 6102
DeinitTrailing ... 6103
DeinitMoney ... 6104
DeinitIndicators ... 6105
Refresh ... 6106
Processing ... 6107
SelectPosition ... 6109
CheckOpen ... 6110
CheckOpenLong ... 6111
CheckOpenShort ... 6112
OpenLong ... 6113
OpenShort ... 6114
CheckReverse ... 6115
CheckReverseLong ... 6116
CheckReverseShort ... 6117
ReverseLong ... 6118
ReverseShort ... 6119
CheckClose ... 6120
CheckCloseLong ... 6121
CheckCloseShort ... 6122
CloseAll ... 6123
Close ... 6124
CloseLong ... 6125
CloseShort ... 6126
CheckTrailingStop ... 6127
CheckTrailingStopLong ... 6128
CheckTrailingStopShort ... 6129
TrailingStopLong ... 6130
TrailingStopShort ... 6131
CheckTrailingOrderLong ... 6132
CheckTrailingOrderShort ... 6133
TrailingOrderLong ... 6134
TrailingOrderShort ... 6135
CheckDeleteOrderLong ... 6136
CheckDeleteOrderShort ... 6137
DeleteOrders ... 6138
DeleteOrder ... 6139
DeleteOrderLong ... 6140
DeleteOrderShort ... 6141
LotOpenLong ... 6142
LotOpenShort ... 6143
LotReverse ... 6144
PrepareHistoryDate ... 6145
HistoryPoint ... 6146
CheckTradeState ... 6147
WaitEvent ... 6148

© 2000-2025, MetaQuotes Ltd.

NoWaitEvent ... 6149
TradeEventPositionStopTake ... 6150
TradeEventOrderTriggered ... 6151
TradeEventPositionOpened ... 6152
TradeEventPositionVolumeChanged ... 6153
TradeEventPositionModified ... 6154
TradeEventPositionClosed ... 6155
TradeEventOrderPlaced ... 6156
TradeEventOrderModified ... 6157
TradeEventOrderDeleted ... 6158
TradeEventNotIdentified ... 6159
TimeframeAdd ... 6160
TimeframesFlags ... 6161
CExpertSignal ... 6162
BasePrice ... 6165
UsedSeries ... 6166
Weight ... 6167
PatternsUsage ... 6168
General ... 6169
Ignore ... 6170
Invert ... 6171
ThresholdOpen ... 6172
ThresholdClose ... 6173
PriceLevel ... 6174
StopLevel ... 6175
TakeLevel ... 6176
Expiration ... 6177
Magic ... 6178
ValidationSettings ... 6179
InitIndicators ... 6180
AddFilter ... 6181
CheckOpenLong ... 6182
CheckOpenShort ... 6183
OpenLongParams ... 6184
OpenShortParams ... 6185
CheckCloseLong ... 6186
CheckCloseShort ... 6187
CloseLongParams ... 6188
CloseShortParams ... 6189
CheckReverseLong ... 6190
CheckReverseShort ... 6191
CheckTrailingOrderLong ... 6192
CheckTrailingOrderShort ... 6193
LongCondition ... 6194
ShortCondition ... 6195
Direction ... 6196
CExpertTrailing ... 6197
CheckTrailingStopLong ... 6199
CheckTrailingStopShort ... 6200
CExpertMoney ... 6201
Percent ... 6203
ValidationSettings ... 6204
CheckOpenLong ... 6205
CheckOpenShort ... 6206
CheckReverse ... 6207
CheckClose ... 6208
Módulos de señales de trading ... 6209
Signals of the Indicator Accelerator Oscillator ... 6212

© 2000-2025, MetaQuotes Ltd.

Signals of the Indicator Adaptive Moving Average 6215
Signals of the Indicator Awesome Oscillator 6219
Signals of the Oscillator Bears Power 6223
Signals of the Oscillator Bulls Power 6225
Signals of the Oscillator Commodity Channel Index 6227
Signals of the Oscillator DeMarker 6231
Signals of the Indicator Double Exponential Moving Average 6235
Signals of the Indicator Envelopes 6239
Signals of the Indicator Fractal Adaptive Moving Average 6242
Signals of the Intraday Time Filter 6246
Signals of the Oscillator MACD 6248
Signals of the Indicator Moving Average 6254
Signals of the Indicator Parabolic SAR 6258
Signals of the Oscillator Relative Strength Index 6260
Signals of the Oscillator Relative Vigor Index 6266
Signals of the Oscillator Stochastic 6268
Signals of the Oscillator Triple Exponential Average 6273
Signals of the Indicator Triple Exponential Moving Average 6277
Signals of the Oscillator Williams Percent Range 6281
Trailing Stop Classes 6284
CTrailingFixedPips 6285
StopLevel 6287
ProfitLevel 6288
ValidationSettings 6289
CheckTrailingStopLong 6290
CheckTrailingStopShort 6291
CTrailingMA 6292
Period 6294
Shift 6295
Method 6296
Applied 6297
InitIndicators 6298
ValidationSettings 6299
CheckTrailingStopLong 6300
CheckTrailingStopShort 6301
CTrailingNone 6302
CheckTrailingStopLong 6303
CheckTrailingStopShort 6304
CTrailingPSAR 6305
Step 6307
Maximum 6308
InitIndicators 6309
CheckTrailingStopLong 6310
CheckTrailingStopShort 6311
Clases de gestión del dinero 6312
CMoneyFixedLot 6313
Lots 6314
ValidationSettings 6315
CheckOpenLong 6316
CheckOpenShort 6317
CMoneyFixedMargin 6318
CheckOpenLong 6319
CheckOpenShort 6320
CMoneyFixedRisk 6321
CheckOpenLong 6322
CheckOpenShort 6323
CMoneyNone 6324
ValidationSettings 6325

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong ... 6326
CheckOpenShort ... 6327
CMoneySizeOptimized ... 6328
DecreaseFactor ... 6330
ValidationSettings ... 6331
CheckOpenLong ... 6332
CheckOpenShort ... 6333
Paneles y ventanas de diálogo ... 6334
CRect ... 6336
Left ... 6337
Top ... 6338
Right ... 6339
Bottom ... 6340
Width ... 6341
Height ... 6342
SetBound ... 6343
Move ... 6344
Shift ... 6345
Contains ... 6346
Format ... 6347
CDateTime ... 6348
MonthName ... 6350
ShortMonthName ... 6351
DayName ... 6352
ShortDayName ... 6353
DaysInMonth ... 6354
DateTime ... 6355
Date ... 6356
Time ... 6357
Sec ... 6358
Min ... 6359
Hour ... 6360
Day ... 6361
Mon ... 6362
Year ... 6363
SecDec ... 6364
SecInc ... 6365
MinDec ... 6366
MinInc ... 6367
HourDec ... 6368
HourInc ... 6369
DayDec ... 6370
DayInc ... 6371
MonDec ... 6372
MonInc ... 6373
YearDec ... 6374
YearInc ... 6375
CWnd ... 6376
Create ... 6380
Destroy ... 6381
OnEvent ... 6382
OnMouseEvent ... 6383
Name ... 6384
ControlsTotal ... 6385
Control ... 6386
ControlFind ... 6387
Rect ... 6388
Left ... 6389

© 2000-2025, MetaQuotes Ltd.

Top ... 6390
Right ... 6391
Bottom ... 6392
Width ... 6393
Height ... 6394
Move ... 6395
Shift ... 6396
Resize ... 6397
Contains ... 6398
Alignment ... 6399
Align ... 6400
Id ... 6401
IsEnabled ... 6402
Enable ... 6403
Disable ... 6404
IsVisible ... 6405
Visible ... 6406
Show ... 6407
Hide ... 6408
IsActive ... 6409
Activate ... 6410
Deactivate ... 6411
StateFlags ... 6412
StateFlagsSet ... 6413
StateFlagsReset ... 6414
PropFlags ... 6415
PropFlagsSet ... 6416
PropFlagsReset ... 6417
MouseX ... 6418
MouseY ... 6419
MouseFlags ... 6420
MouseFocusKill ... 6421
OnCreate ... 6422
OnDestroy ... 6423
OnMove ... 6424
OnResize ... 6425
OnEnable ... 6426
OnDisable ... 6427
OnShow ... 6428
OnHide ... 6429
OnActivate ... 6430
OnDeactivate ... 6431
OnClick ... 6432
OnChange ... 6433
OnMouseDown ... 6434
OnMouseUp ... 6435
OnDragStart ... 6436
OnDragProcess ... 6437
OnDragEnd ... 6438
DragObjectCreate ... 6439
DragObjectDestroy ... 6440
CWndObj ... 6441
OnEvent ... 6443
Text ... 6444
Color ... 6445
ColorBackground ... 6446
ColorBorder ... 6447
Font ... 6448

© 2000-2025, MetaQuotes Ltd.

FontSize ... 6449
ZOrder ... 6450
OnObjectCreate ... 6451
OnObjectChange ... 6452
OnObjectDelete ... 6453
OnObjectDrag ... 6454
OnSetText ... 6455
OnSetColor ... 6456
OnSetColorBackground ... 6457
OnSetFont ... 6458
OnSetFontSize ... 6459
OnSetZOrder ... 6460
OnDestroy ... 6461
OnChange ... 6462
CWndContainer ... 6463
Destroy ... 6465
OnEvent ... 6466
OnMouseEvent ... 6467
ControlsTotal ... 6468
Control ... 6469
ControlFind ... 6470
Add ... 6471
Delete ... 6472
Move ... 6473
Shift ... 6474
Id ... 6475
Enable ... 6476
Disable ... 6477
Show ... 6478
Hide ... 6479
MouseFocusKill ... 6480
Save ... 6481
Load ... 6482
OnResize ... 6483
OnActivate ... 6484
OnDeactivate ... 6485
CLabel ... 6486
Create ... 6491
OnSetText ... 6492
OnSetColor ... 6493
OnSetFont ... 6494
OnSetFontSize ... 6495
OnCreate ... 6496
OnShow ... 6497
OnHide ... 6498
OnMove ... 6499
CBmpButton ... 6500
Create ... 6507
Border ... 6508
BmpNames ... 6509
BmpOffName ... 6510
BmpOnName ... 6511
BmpPassiveName ... 6512
BmpActiveName ... 6513
Pressed ... 6514
Locking ... 6515
OnSetZOrder ... 6516
OnCreate ... 6517

© 2000-2025, MetaQuotes Ltd.

OnShow ... 6518
OnHide ... 6519
OnMove ... 6520
OnChange ... 6521
OnActivate ... 6522
OnDeactivate ... 6523
OnMouseDown ... 6524
OnMouseUp ... 6525
CButton ... 6526
Create ... 6533
Pressed ... 6534
Locking ... 6535
OnSetText ... 6536
OnSetColor ... 6537
OnSetColorBackground ... 6538
OnSetColorBorder ... 6539
OnSetFont ... 6540
OnSetFontSize ... 6541
OnCreate ... 6542
OnShow ... 6543
OnHide ... 6544
OnMove ... 6545
OnResize ... 6546
OnMouseDown ... 6547
OnMouseUp ... 6548
CEdit ... 6549
Create ... 6555
ReadOnly ... 6556
TextAlign ... 6557
OnObjectEndEdit ... 6558
OnSetText ... 6559
OnSetColor ... 6560
OnSetColorBackground ... 6561
OnSetColorBorder ... 6562
OnSetFont ... 6563
OnSetFontSize ... 6564
OnSetZOrder ... 6565
OnCreate ... 6566
OnShow ... 6567
OnHide ... 6568
OnMove ... 6569
OnResize ... 6570
OnChange ... 6571
OnClick ... 6572
CPanel ... 6573
Create ... 6578
BorderType ... 6579
OnSetText ... 6580
OnSetColorBackground ... 6581
OnSetColorBorder ... 6582
OnCreate ... 6583
OnShow ... 6584
OnHide ... 6585
OnMove ... 6586
OnResize ... 6587
OnChange ... 6588
CPicture ... 6589
Create ... 6594

© 2000-2025, MetaQuotes Ltd.

Border ... 6595
BmpName ... 6596
OnCreate ... 6597
OnShow ... 6598
OnHide ... 6599
OnMove ... 6600
OnChange ... 6601
CScroll ... 6602
Create ... 6604
OnEvent ... 6605
MinPos ... 6606
MaxPos ... 6607
CurrPos ... 6608
CreateBack ... 6609
CreateInc ... 6610
CreateDec ... 6611
CreateThumb ... 6612
OnClickInc ... 6613
OnClickDec ... 6614
OnShow ... 6615
OnHide ... 6616
OnChangePos ... 6617
OnThumbDragStart ... 6618
OnThumbDragProcess ... 6619
OnThumbDragEnd ... 6620
CalcPos ... 6621
CScrollV ... 6622
CreateInc ... 6628
CreateDec ... 6629
CreateThumb ... 6630
OnResize ... 6631
OnChangePos ... 6632
OnThumbDragStart ... 6633
OnThumbDragProcess ... 6634
OnThumbDragEnd ... 6635
CalcPos ... 6636
CScrollH ... 6637
CreateInc ... 6643
CreateDec ... 6644
CreateThumb ... 6645
OnResize ... 6646
OnChangePos ... 6647
OnThumbDragStart ... 6648
OnThumbDragProcess ... 6649
OnThumbDragEnd ... 6650
CalcPos ... 6651
CWndClient ... 6652
Create ... 6655
OnEvent ... 6656
ColorBackground ... 6657
ColorBorder ... 6658
BorderType ... 6659
VScrolled ... 6660
HScrolled ... 6661
CreateBack ... 6662
CreateScrollV ... 6663
CreateScrollH ... 6664
OnResize ... 6665

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow ... 6666
OnVScrollHide ... 6667
OnHScrollShow ... 6668
OnHScrollHide ... 6669
OnScrollLineDown ... 6670
OnScrollLineUp ... 6671
OnScrollLineLeft ... 6672
OnScrollLineRight ... 6673
Rebound ... 6674
CListView ... 6675
Create ... 6681
OnEvent ... 6682
TotalView ... 6683
AddItem ... 6684
Select ... 6685
SelectByText ... 6686
SelectByValue ... 6687
Value ... 6688
CreateRow ... 6689
OnResize ... 6690
OnVScrollShow ... 6691
OnVScrollHide ... 6692
OnScrollLineDown ... 6693
OnScrollLineUp ... 6694
OnItemClick ... 6695
Redraw ... 6696
RowState ... 6697
CheckView ... 6698
CComboBox ... 6699
Create ... 6705
OnEvent ... 6706
AddItem ... 6707
ListViewItems ... 6708
Select ... 6709
SelectByText ... 6710
SelectByValue ... 6711
Value ... 6712
CreateEdit ... 6713
CreateButton ... 6714
CreateList ... 6715
OnClickEdit ... 6716
OnClickButton ... 6717
OnChangeList ... 6718
ListShow ... 6719
ListHide ... 6720
CCheckBox ... 6721
Create ... 6727
OnEvent ... 6728
Text ... 6729
Color ... 6730
Checked ... 6731
Value ... 6732
CreateButton ... 6733
CreateLabel ... 6734
OnClickButton ... 6735
OnClickLabel ... 6736
CCheckGroup ... 6737
Create ... 6743

© 2000-2025, MetaQuotes Ltd.

OnEvent ... 6744
AddItem ... 6745
Value ... 6746
CreateButton ... 6747
OnVScrollShow ... 6748
OnVScrollHide ... 6749
OnScrollLineDown ... 6750
OnScrollLineUp ... 6751
OnChangeItem ... 6752
Redraw ... 6753
RowState ... 6754
CRadioButton ... 6755
Create ... 6757
OnEvent ... 6758
Text ... 6759
Color ... 6760
State ... 6761
CreateButton ... 6762
CreateLabel ... 6763
OnClickButton ... 6764
OnClickLabel ... 6765
CRadioGroup ... 6766
Create ... 6772
OnEvent ... 6773
AddItem ... 6774
Value ... 6775
CreateButton ... 6776
OnVScrollShow ... 6777
OnVScrollHide ... 6778
OnScrollLineDown ... 6779
OnScrollLineUp ... 6780
OnChangeItem ... 6781
Redraw ... 6782
RowState ... 6783
Select ... 6784
CSpinEdit ... 6785
Create ... 6791
OnEvent ... 6792
MinValue ... 6793
MaxValue ... 6794
Value ... 6795
CreateEdit ... 6796
CreateInc ... 6797
CreateDec ... 6798
OnClickInc ... 6799
OnClickDec ... 6800
OnChangeValue ... 6801
CDialog ... 6802
Create ... 6804
OnEvent ... 6805
Caption ... 6806
Add ... 6807
CreateWhiteBorder ... 6808
CreateBackground ... 6809
CreateCaption ... 6810
CreateButtonClose ... 6811
CreateClientArea ... 6812
OnClickCaption ... 6813

© 2000-2025, MetaQuotes Ltd.

OnClickButtonClose ... 6814
ClientAreaVisible ... 6815
ClientAreaLeft ... 6816
ClientAreaTop ... 6817
ClientAreaRight ... 6818
ClientAreaBottom ... 6819
ClientAreaWidth ... 6820
ClientAreaHeight ... 6821
OnDialogDragStart ... 6822
OnDialogDragProcess ... 6823
OnDialogDragEnd ... 6824
CAppDialog ... 6825
Create ... 6828
Destroy ... 6829
OnEvent ... 6830
Run ... 6831
ChartEvent ... 6832
Minimized ... 6833
IniFileSave ... 6834
IniFileLoad ... 6835
IniFileName ... 6836
IniFileExt ... 6837
CreateCommon ... 6838
CreateExpert ... 6839
CreateIndicator ... 6840
CreateButtonMinMax ... 6841
OnClickButtonClose ... 6842
OnClickButtonMinMax ... 6843
OnAnotherApplicationClose ... 6844
Rebound ... 6845
Minimize ... 6846
Maximize ... 6847
CreateInstanceId ... 6848
ProgramName ... 6849
SubwinOff ... 6850
36 Paso de la versión MQL4 ... 6851
37 Lista de Funciones MQL5 ... 6855
38 Lista de Constantes MQL5 ... 6892

© 2000-2025, MetaQuotes Ltd.

Manual de referencia MQL5

MetaQuotes Language 5 (MQL5) es un lenguaje de programación de indicadores técnicos, robots comerciales y aplicaciones auxiliares para automatizar el comercio en los mercados financieros. MQL5 es un lenguaje moderno de alto nivel, desarrollado por MetaQuotes para su propia plataforma comercial e informativa. La sintaxis es muy semejante a C++ y permite escribir programas en el estilo de la programación orientada a objetos (POO).

Para escribir programas en MQL5, dentro de la plataforma comercial se ofrece el entorno de desarrollo MetaEditor con todas las herramientas modernas necesarias para escribir código, incluyendo plantillas, snipetts, depuración, perfilado, autocompletar y el repositorio de versiones incorporado MQL5 Storage.

El soporte y el desarrollo del lenguaje tiene lugar en el sitio web MQL5 Algotrading community, donde podrá encontrar una amplia bibilioteca de códigos gratuitos y multitud de artículos. Estos artículos abarcan todos los temas del trading moderno: redes neuronales, estadística y análisis, trading de alta frecuencia, arbitraje, simulación y optimización de estrategias comerciales, uso de robots para automatizar el comercio y mucho más.

Los tráders y desarrolladores de programas MQL5 pueden comunicarse en el foro, realizar encargos en Freelance y comprar y vender programas protegidos en el Mercado, la tienda de aplicaciones para trading ya preparadas.

El lenguaje MQL5 contiene funciones comerciales especializadas y procesadores de eventos predeterminados para escribir asesores expertos (Expert Advisors). Los asesores gestionan automáticamente los procesos comerciales basándose en las reglas comerciales implementadas en ellos. Asimismo, en MQL5 es posible crear indicadores técnicos propios (Custom Indicators), scrips y bibliotecas de funciones (Libraries).

El manual de referencia de MQL5 contiene funciones, operaciones, palabras reservadas y otras construcciones del lenguaje, todo dividido en categorías. Asimismo, permite conocer la descripción de cada elemento usado e incluido en el lenguaje. Además, en el manual de referencia, se muestra la descripción de las clases de la Biblioteca estándar, para crear estrategias comerciales, paneles de control, gráficos personalizados y trabajos con archivos.

Aparte del manual de referencia de CodeBase, se ha publicado la biblioteca de análisis numérico ALGLIB, que permite resolver multitud de tareas matemáticas.

Manual sobre trading algorítmico

Empezar a aprender algo nuevo nunca es fácil. Para ayudar a los principiantes, hemos publicado dos libros de texto sobre programación en el lenguaje MQL5, destinados a todos aquellos que quieren dominar la creación de robots y aplicaciones comerciales para el trading algorítmico.

Nuestros libros de texto ofrecen a los lectores una presentación sistemática y estructurada del material, lo cual facilita enormemente el proceso de aprendizaje de MQL5. Los ejemplos de código detallados, que explican paso a paso la creación de aplicaciones y robots comerciales, le permitirán comprender y dominar mejor las complejidades del comercio algorítmico. Los libros contienen muchos ejercicios prácticos que le ayudarán a consolidar sus conocimientos y desarrollar habilidades de programación en condiciones comerciales reales.

© 2000-2025, MetaQuotes Ltd.

"Programación en MQL5 para tráders" es el libro de texto más completo y detallado sobre MQL5, adecuado para programadores de cualquier nivel. Los principiantes podrán familiarizarse con los fundamentos del trading: desde el estudio de herramientas de desarrollo hasta los conceptos básicos de programación. Asimismo, tendrán la posibilidad de crear su primer programa, compilarlo y ejecutarlo en la plataforma comercial MetaTrader 5. Los usuarios con experiencia en programación en otros lenguajes podrán pasar directamente a la parte aplicada: las capacidades de MQL5 para crear robots comerciales y aplicaciones analíticas.

"Redes neuronales en el trading algorítmico en MQL5" es un tutorial sobre el uso de métodos de aprendizaje automático para crear robots comerciales en la plataforma MetaTrader 5. Poco a poco se familiarizará con los conceptos básicos de las redes neuronales y sus posibilidades de uso en el trading algorítmico. A medida que avance, irá creando y entrenando su propia inteligencia artificial, añadiéndole poco a poco nuevas funciones. Además de estudiar MQL5, usted aprenderá a escribir programas en Python y OpenCL, y también aprenderá sobre métodos matriciales y vectoriales integrados que le permitirán resolver problemas matemáticos complejos utilizando un código conciso y rápido.

Artículos sobre el desarrollo de aplicaciones comerciales

Una excelente guía para aprender todas las capacidades del lenguaje son los artículos sobre MQL5 en los que se analizan muchos problemas prácticos en el trading algorítmico. Para facilitar la búsqueda, todos los artículos se dividen por categorías: Ejemplos, Expertos, Aprendizaje automático, etc. Cada mes, se publican docenas de nuevos artículos en el sitio web MQL5 Algotrading community, escritos por los propios tráders para otros tráders. Lea y analice los artículos: en ellos se discuten todos los aspectos del trading algorítmico moderno. Para los principiantes, hemos compilado una lista de 16 artículos recomendados para una inmersión rápida en MQL5.

Tipos de aplicaciones en MQL5

Para ejecutar tareas concretas de automatización de operaciones comerciales, los programas MQL5 se dividen en cinco tipos especializados:

- **Asesor**: sistema comercial automático, vinculado a un gráfico determinado. El asesor contiene funciones-manejadores de eventos predeterminados; cuando estos tienen lugar, se ejecutan los elementos correspondientes de la estrategia comercial. Ejemplos de estos eventos: inicialización y desinicialización del programa, llegada de un nuevo tick, activación del temporizador, cambio en la profundidad del mercado, eventos personalizados.
  El asesor no solo puede calcular señales comerciales según las reglas implementadas, sino también ejecutar transacciones automáticamente en la cuenta comercial, dirigiéndolas directamente al servidor comercial. El asesor se guarda en el directorio *<catálogo_del_terminal>\MQL5\Experts*.
- **Indicador personalizado**: indicador técnico escrito por el usuario en adición a los indicadores ya integrados en la plataforma comercial. Los indicadores personalizados, al igual que los incorporados, no pueden comerciar automáticamente y están pensados solo para implementar funciones analíticas. Los indicadores personalizados pueden utilizar en sus cálculos los valores de otros indicadores, además, también se pueden llamar en asesores.
  Los indicadores personalizados se guardan en el directorio *<catálogo_del_terminal>\MQL5\Indicators*.

© 2000-2025, MetaQuotes Ltd.

- **Script**: programa diseñado para la ejecución única de algunas acciones. A diferencia de los expertos, los scripts no procesan ningún evento, aparte de los eventos de inicio, inicialización y desinicialización. Para que el script funcione, en su código deberá estar la función-manejador OnStart. Los scripts se guardan en el directorio <*catálogo_del_terminal*>*MQL5**Scripts*.
- **Servicio**: programa que, a diferencia de los asesores, indicadores y scripts, no requiere estar vinculado a un gráfico para funcionar. Al igual que los scripts, los servicios no procesan ningún evento, aparte del evento de inicio. Para iniciar un servicio, en su código deberá estar la función-manejador OnStart. Los servicios no aceptan ningún evento excepto Start, aunque pueden enviar por sí mismos a los gráficos eventos personalizados con la ayuda de EventChartCustom. Los servicios se guardan en el directorio <*catálogo_del_terminal*>*MQL5**Services*.
- **Biblioteca**: biblioteca de funciones personalizadas, diseñada para guardar y distribuir los bloques más utilizados de los programas personalizados. Las bibliotecas no pueden iniciarse para su ejecución de forma autónoma.
  Las bibliotecas se guardan en el directorio <*catálogo_del_terminal*>*MQL5**Libraries*
- **Archivo de inclusión**: código fuente de bloques (usados con frecuencia) de programas personalizados. Estos archivos pueden incluirse en los códigos fuente de los expertos, scripts, indicadores personalizados y bibliotecas en la etapa de compilación. Los archivos de inclusión utilizados son preferibles a las bibliotecas personalizadas, debido a la sobrecarga adicional que supone llamar a las funciones de las bibliotecas.
  Los archivos de inclusión pueden encontrarse en el mismo directorio que el archivo fuente, en este caso, se usa la directiva #include con comillas dobles. Otro lugar de guardado de los archivos de inclusión es el directorio <*catálogo_del_terminal*>*MQL5**Include*, en este caso, se usa la directiva *#include* con paréntesis angulares.

© 2000-2025, MetaQuotes Ltd

Bases del lenguaje

El lenguaje MetaQuotes Language 5 (MQL5) es un lenguaje de programación orientado a objetos de alto nivel y está destinado para diseñar las estrategias automáticas de trading, indicadores técnicos personalizados con el fin de analizar diferentes mercados financieros. No sólo permite diseñar diferentes sistemas expertos destinados para trabajar en tiempo real sino crear sus propias herramientas gráficas que ayudan a tomar decisiones comerciales.

MQL5 se basa en concepto del lenguaje de programación muy usual C++. En comparación con MQL4, han sido agregadas las enumeraciones, estructuras, clases y procesamiento de eventos. La interacción de los programas procesados en MQL5 con las demás aplicaciones mediante dll ha sido facilitado al máximo gracias al aumento de cantidad de tipos principales incorporados. La sintaxis del lenguaje MQL5 es parecida a la del C++, lo que permite traspasar a él sin ninguna dificultad los programas escritos en otros modernos lenguajes de programación.

Con el fin de aprender el lenguaje MQL5 todos los temas han sido agrupados en los siguientes apartados:

- Sintaxis
- Tipos de datos
- Operaciones y expresiones
- Operadores
- Funciones
- Variables
- Preprocesador
- Programación orientada a objetos
- Espacios de nombres

© 2000-2025, MetaQuotes Ltd.

Sintaxis

En aspecto de la sintaxis el lenguaje de programación de estrategias comerciales MQL5 es muy parecido al lenguaje de programación C++, salvo algunas posibilidades:

- falta la aritmética de dirección;
- falta el operador goto;
- no se puede utilizar la enumeración anónima;
- no hay herencia múltiple.

Véase también

Enumeraciones, Estructuras y clases, Herencia

© 2000-2025, MetaQuotes Ltd.

Comentarios

Los comentarios en bloque se empiezan con los símbolos /* y se terminan con */. Estos comentarios no pueden ser insertados. Los comentarios en línea se empiezan con los símbolos //, se terminan con el símbolo de nueva línea, se puede insertarlos en los comentarios en bloque. Esta permitido el uso de los comentarios en cualquier lugar donde se puede utilizar los espacios, además la cantidad de espacios no está limitada.

Ejemplos:

```
//--- Comentario en línea
/*  Comentario
    en          // Comentario en línea insertado
    bloque
*/
```

© 2000-2025, MetaQuotes Ltd.

Identificadores

Los identificadores se utilizan como los nombres para las variables y funciones. Los identificadores no pueden tener más de 63 caracteres.

Los símbolos permitidos para escribir los identificadores son los siguientes: dígitos numéricos de 0 a 9, letras mayúsculas y minúsculas latinas a-z y A-Z se reconocen como símbolos distintos, guión bajo (_). Dígitos numéricos en ningún caso deben aparecer como primer carácter.

Un identificador no puede coincidir con una palabra reservada.

Ejemplos:

```
NAME1 name1 Total_5 Paper
```

Véase también

Variables, Funciones

© 2000-2025, MetaQuotes Ltd.

Palabras reservadas

Los identificadores especificados más abajo se determinan como palabras reservadas a cada una de las cuales le corresponde una acción determinada, y no pueden ser utilizados en otro sentido:

Tipos de datos

bool	float	uint
char	int	ulong
class	long	union
color	short	ushort
datetime	string	void
double	struct	
enum	uchar	

Especificadores de acceso

const	private	virtual
delete	protected	
override	public	

Clases de memoria

extern	input	static

Operadores

break	dynamic_cast	operator
case	else	pack
continue	for	return
default	if	sizeof
delete	new	switch
do	offsetof	while

Otros

this	#define	#import
true	#ifdef	#include

© 2000-2025, MetaQuotes Ltd.

this	#define	#import
false	#ifndef	#property
template	#else	group
typename	#endif	namespace

© 2000-2025, MetaQuotes Ltd.

Tipos de datos

Cualquier programa opera con los los datos. Dependiendo de su función, éstos pueden ser de varios tipos. Por ejemplo, para acceder a los elementos de un array se utilizan los datos del tipo de números enteros. Los datos de precios tienen el tipo de doble precisión en punto flotante. Esto se debe a que en el lenguaje MQL5 no está previsto ningún tipo especial para los datos de precios.

Los datos de diferentes tipos se procesan a diferentes velocidades. El procesamiento de datos de números enteros se realiza más rápido. Para procesar datos de doble precisión se utiliza un coprocesador especial. No obstante, a causa de la complejidad de demostración interna de los datos de punto flotante, éstos requieren más tiempo para ser procesados que los de números enteros.

Los que más lentamente se procesan son los datos en cadena de caracteres. Esto está relacionado con la distribución y redistribución dinámica de la memoria operativa del ordenador.

Principales tipos de datos:

- enteros (char, short, int, long, uchar, ushort, uint, ulong)
- lógicos (bool)
- literales (char, uchar)
- cadenas de caracteres (string)
- de punto flotante (double, float)
- color (color)
- fecha y tiempo (datetime)
- enumeraciones (enum)

Tipos de datos compuestos:

- estructuras;
- clases.

En los términos de POO los tipos complejos de datos obtienen el nombre de los tipos abstractos de datos.

Los tipos color y datetime sirven unicamente para la comodidad de presentación e introducción de los parámetros hechos desde fuera: la tabla de propiedades del asesor o indicador del usuario (pestaña "Inputs"). Los datos de los tipos color y datetime se representan como los números enteros. Los tipos enteros y los del punto flotante se llaman los tipos aritméticos (numéricos).

En las expresiones se utiliza la conversión implícita de tipos si no se indica lo contario (la explícita).

Véase también

Conversión de tipos

© 2000-2025, MetaQuotes Ltd.

Tipos enteros

En el lenguaje MQL5 los tipos enteros están representados por once categorías. Algunos de ellos se puede utilizar en conjunto con los demás si lo requiere la lógica del programa, aunque en este caso hay que tener en cuenta las normas de conversión de tipos.

En la tabla de abajo se muestran las características de cada tipo. Además, en la última columna, para cada tipo se indica su análogo en el lenguaje de programación C++.

Tipo	Tamaño en bytes	Valor mínimo	Valor máximo	Análogo en el lenguaje C++
char	1	-128	127	char
uchar	1	0	255	unsigned char, BYTE
bool	1	0(false)	1(true)	bool
short	2	-32 768	32 767	short, wchar_t
ushort	2	0	65 535	unsigned short, WORD
int	4	- 2 147 483 648	2 147 483 647	int
uint	4	0	4 294 967 295	unsigned int, DWORD
color	4	-1	16 777 215	int, COLORREF
long	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	__int64
ulong	8	0	18 446 744 073 709 551 615	unsigned __int64
datetime	8	0 (1970.01.01 0:00:00)	32 535 244 799 (3000.12.31 23:59:59)	__time64_t

Los valores de tipos enteros también pueden ser representados en forma de las constantes numéricas, literales de color, literales de fecha-hora, constantes de signos y enumeraciones.

Pseudónimos adicionales de tipos enteros (para compatibilidad con C/C++)

Para simplificar la portabilidad de código desde otros lenguajes, como C y C++, se han añadido al lenguaje MQL5 pseudónimos para los tipos enteros estándar. Estos pseudónimos no introducen nuevos tipos, sino que suponen nombres alternativos de tipos ya existentes en MQL5. Se pueden utilizar en todos los contextos en los que los tipos básicos son aplicables.

© 2000-2025, MetaQuotes Ltd.

Tipo	Se corresponde con el tipo MQL5	Tamaño (bit)	Signo	Valor mínimo	Valor máximo
int8_t	char	8	signed	-128	127
uint8_t	uchar	8	unsigned	0	255
int16_t	short	16	signed	-32 768	32 767
uint16_t	ushort	16	unsigned	0	65 535
int32_t	int	32	signed	- 2 147 483 648	2 147 483 647
uint32_t	uint	32	unsigned	0	4 294 967 295
int64_t	long	64	signed	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
uint64_t	ulong	64	unsigned	0	18 446 744 073 709 551 615

El uso de estos pseudónimos resulta útil al portar bibliotecas y algoritmos existentes de C/C++ a MQL5, especialmente en proyectos con un alto nivel de compatibilidad de plataformas. Todos los nuevos nombres están disponibles tanto en scripts como en asesores expertos e indicadores.

Véase también

Conversión de datos, Constantes de tipos numéricos

© 2000-2025, MetaQuotes Ltd.

Tipos char, short, int y long

char

El tipo entero char ocupa en la memoria 1 byte (8 bits) y permite representar en el sistema numérico binario 2^8 valores = 256. El tipo char puede contener los valores positivos, igual que negativos. El rango de valores es de -128 a 127.

uchar

El tipo entero uchar también ocupa en la memoria 1 byte, igual que el tipo char, pero a diferencia de él, uchar está destinado únicamente para los valores positivos. El valor mínimo es igual a cero, el valor máximo es igual a 255. La primera letra u del nombre uchar es la abreviatura de la palabra unsigned (sin signo).

short

El tipo entero short tiene el tamaño de 2 bytes (16 bits), permite representar la multitud de valores igual a 2 elevado a 16: 2^16 = 65 536. Puesto que el tipo short es con signos y contiene los valores tanto positivos, como negativos, el rango de valores se oscila entre -32 768 y 32 767.

ushort

El tipo ushort es el tipo short sin signos, también tiene el tamaño de 2 bytes. El valor mínimo es igual a cero, el valor máximo es igual a 65 535.

int

El tipo entero int tiene el tamaño de 4 bytes (32 bits). El valor mínimo es de -2 147 483 648, el valor máximo es de 2 147 483 647.

uint

El tipo entero sin signos uint ocupa en la memoria 4 bytes y permite representar los valores de números enteros de 0 a 4 294 967 295.

long

El tipo entero long tiene el tamaño de 8 bytes (64 bits). El valor mínimo es de -9 223 372 036 854 775 808, el valor máximo es de 9 223 372 036 854 775 807.

ulong

El tipo entero ulong también ocupa 8 bytes y permite almacenar valores de 0 a 18 446 744 073 709 551 615.

Ejemplos:

```
char  ch=12;
```

© 2000-2025, MetaQuotes Ltd.

```
short sh=-5000;
int   in=2445777;
```

Debido a que los tipos enteros sin signos no sirven para almacenar los valores negativos, el intento de poner los valores negativos puede llevar a las consecuencias inesperadas. Pues un script inocente como éste llevará a un ciclo contínuo:

```
//--- ciclo continuo
void OnStart()
  {
   uchar  u_ch;

   for(char ch=-128;ch<128;ch++)
     {
      u_ch=ch;
      Print("ch = ",ch," u_ch = ",u_ch);
     }
  }
```

Así es correcto:

```
//--- versión correcta
void OnStart()
  {
   uchar  u_ch;

   for(char ch=-128;ch<=127;ch++)
     {
      u_ch=ch;
      Print("ch = ",ch," u_ch = ",u_ch);
      if(ch==127) break;
     }
  }
```

Resultado:

```
ch= -128  u_ch= 128
ch= -127  u_ch= 129
ch= -126  u_ch= 130
ch= -125  u_ch= 131
ch= -124  u_ch= 132
ch= -123  u_ch= 133
ch= -122  u_ch= 134
ch= -121  u_ch= 135
```

© 2000-2025, MetaQuotes Ltd.

```
    ch= -120  u_ch= 136

    ch= -119  u_ch= 137

    ch= -118  u_ch= 138

    ch= -117  u_ch= 139

    ch= -116  u_ch= 140

    ch= -115  u_ch= 141

    ch= -114  u_ch= 142

    ch= -113  u_ch= 143

    ch= -112  u_ch= 144

    ch= -111  u_ch= 145

     ...
```

Ejemplos:

```
//--- no se puede guardar los valores negativos en los tipos sin signos
uchar  u_ch=-120;
ushort u_sh=-5000;
uint   u_in=-401280;
```

Hexadecimales: cífras 0-9, letras a-f o A-F para los valores 10-15, se empiezan con 0x o 0X.

Ejemplos:

```
0x0A, 0x12, 0X12, 0x2f, 0xA3, 0Xa3, 0X7C7
```

Para las variables de números enteros, los valores se puede establecer en forma binaria usando el prefijo B. Por ejemplo, se puede codificar las horas de trabajo de la sesión de trading en la variable del tipo int y usar la información sobre ellas según el algoritmo necesario:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- para las horas de trabajo ponemos 1, para las horas no laborales ponemos 0
   int AsianSession   =B'111111111'; // sesión asiática desde 0:00 hasta 9:00
   int EuropeanSession=B'111111111000000000'; // sesión europea  9:00 - 18:00
   int AmericanSession =B'111111110000000000000011'; // sesión americana 16:00 - 02:0
//--- mostramos los valores numéricos de las sesiones
   PrintFormat("Asian session hours as value =%d",AsianSession);
   PrintFormat("European session hours as value is %d",EuropeanSession);
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("American session hours as value is %d",AmericanSession);
//--- ahora mostramos las representaciones de cadenas de las horas de trabajo de las s
   Print("Asian session ",GetHoursForSession(AsianSession));
   Print("European session ",GetHoursForSession(EuropeanSession));
   Print("American session ",GetHoursForSession(AmericanSession));
//---
  }
//+------------------------------------------------------------------+
//| devuelve las horas de trabajo de las sesiones en forma de cadena
//+------------------------------------------------------------------+
string GetHoursForSession(int session)
  {
//--- para comprobar, utilizamos las operaciones de bit AND y desplazamiento a un bit
//--- empezamos a comprobar desde el menor bit
   int bit=1;
   string out="working hours: ";
//--- vamos a comprobar todos los 24 bits empezando desde 23 inclusive
   for(int i=0;i<24;i++)
     {
      //--- obtenemos el estado del bit bit en el número number
      bool workinghour=(session&bit)==bit;
      //--- agregamos el número de la hora en el mensaje
      if(workinghour )out=out+StringFormat("%d ",i);
      //--- desplazamos a un bit a la izquierda para comprobar el siguiente
      bit<<=1;
     }
//--- cadena formada
   return out;
  }
```

Véase también

Conversión de tipos

© 2000-2025, MetaQuotes Ltd.

Constantes de caracteres

En MQL5 los caracteres, como elementos de una línea, son los índices en conjunto de caracteres Unicode. Estos son unos valores hexadecimales que pueden ser convertidos en números enteros y con los que se puede ejecutar operaciones de números enteros, tales como, la suma y la resta.

Cualquier carácter solitario que viene entre comillas simples o el código ASCII hexadecimal en forma de '\x10' es una constante de caracteres y tiene el tipo ushort. Por ejemplo, la introducción del '0' representa el valor numérico 30 que corresponde al índice según el cual en la tabla de caracteres se encuentra el signo cero.

Ejemplo:

```
void OnStart()
  {
//--- determinemos las constantes de caracteres
   int symbol_0='0';
   int symbol_9=symbol_0+9; // obtenemos el signo '9'
//--- saquemos los valores de las constantes
   printf("En forma decimal: symbol_0 = %d,  symbol_9 = %d",symbol_0,symbol_9);
   printf("En forma hexadecimal: symbol_0 = 0x%x,  symbol_9 = 0x%x",symbol_0,symbol_9)
//--- introduzcamos las constantes en una línea
   string test="";
   StringSetCharacter(test,0,symbol_0);
   StringSetCharacter(test,1,symbol_9);
//--- así se ven en la línea
   Print(test);
  }
```

Para un compilador durante el análisis de las líneas de constantes y las constantes de caracteres en el código fuente del programa la barra inversa es un carácter controlador. Algunos símbolos, por ejemplo, comillas simples ('), comillas dobles ("), barra inversa (\) y caracteres controladores se puede representar mediante una combinación de signos que se empiezan con la barra inversa (\), según se indica en la tabla de abajo:

Nombre del signo	Código mnemotécnico o imagen	Inscripción en MQL5	Valor numérico
nueva línea (cambio de línea)	LF	'\n'	10
tabulación horizontal	HT	'\t'	9
retorno de carro	CR	'\r'	13
barra inversa	\	'\\'	92
comilla simple	'	'\''	39
comilla doble	"	'\"'	34
código hexadecimal	hhhh	'\xhhhh'	de 1 a 4 símbolos hexadecimales
código decimal	d	'\d'	número decimal de 0 a 65535

© 2000-2025, MetaQuotes Ltd.

Si después de la barra inversa sigue un signo distinto a los arriba mencionados, entonces el resultado no está determinado.

Ejemplo:

```
void OnStart()
  {
//--- declaremos las constantes de caracteres
   int a='A';
   int b='$';
   int c='©';       // código 0xA9
   int d='\xAE';    // código del signo ®
//--- imprimamos las constantes
   Print(a,b,c,d);
//--- agreguemos el signo a la línea
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- cambiemos el signo en la línea
   StringSetCharacter(test,0,b);
   Print(test);
//--- cambiemos el signo en la línea
   StringSetCharacter(test,0,c);
   Print(test);
//--- cambiemos el signo en la línea
   StringSetCharacter(test,0,d);
   Print(test);
//--- representemos signos en forma numérica
   int a1=65;
   int b1=36;
   int c1=169;
   int d1=174;
//--- agreguemos el signo a la línea
   StringSetCharacter(test,1,a1);
   Print(test);
//--- cambiemos el signo en la línea
   StringSetCharacter(test,1,b1);
   Print(test);
//--- cambiemos el signo en la línea
   StringSetCharacter(test,1,c1);
   Print(test);
//--- cambiemos el signo en la línea
   StringSetCharacter(test,1,d1);
   Print(test);
  }
```

Como se decía antes, el valor de una constante de caracteres (o una variable) representa un índice en la tabla de signos, y debido a que éste es un número entero, se permite escribirlo de varias maneras.

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//---
   int a=0xAE;      // código del signo ® corresponde al literal '\xAE'
   int b=0x24;      // código del signo $ corresponde al literal '\x24'
   int c=0xA9;      // código del signo © corresponde al literal '\xA9'
   int d=0x263A;    // código del signo ☺ corresponde al literal '\x263A'
//--- saquemos los valores
   Print(a,b,c,d);
//--- agreguemos el signo a la línea
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- cambiemos el signo en la línea
   StringSetCharacter(test,0,b);
   Print(test);
//--- cambiemos el signo en la línea
   StringSetCharacter(test,0,c);
   Print(test);
//--- cambiemos el signo en la línea
   StringSetCharacter(test,0,d);
   Print(test);
//--- códigos de palos
   int a1=0x2660;
   int b1=0x2661;
   int c1=0x2662;
   int d1=0x2663;
//--- agreguemos el signo de pica
   StringSetCharacter(test,1,a1);
   Print(test);
//--- agreguemos el signo de corazón
   StringSetCharacter(test,2,b1);
   Print(test);
//--- agreguemos el signo de diamante
   StringSetCharacter(test,3,c1);
   Print(test);
//--- agreguemos el signo de trébol
   StringSetCharacter(test,4,d1);
   Print(test);
//--- Ejemplo de literales de signos en la línea
   test="Dama\x2660As\x2662";
   printf("%s",test);
  }
```

La representación interna del literal de signo es el tipo ushort. Las constantes de caracteres pueden adquirir valores de 0 a 65535.

Véase también

StringSetCharacter(), StringGetCharacter(), ShortToString(), ShortArrayToString(), StringToShortArray()

© 2000-2025, MetaQuotes Ltd.

Tipo datetime

El tipo datetime sirve para almacenar la fecha y la hora en forma de cantidad de segundos que han pasado desde el 1 de enero de 1970. Ocupa en la memoria 8 bytes.

Las constantes de fecha y hora pueden estar representadas por una línea de caracteres compuesta de 6 partes que representan el valor numérico del año, mes, día (o día, mes, año), hora, minuto y segundo. La constante se mete entre comillas simples y se empieza con el signo D.

El rango de valores es desde el 1 de enero de 1970 hasta el 31 de diciembre de 3000. Se puede omitir o la fecha (año, mes, día) o la hora (hora, minuto, segundo), o pueden ser las dos cosas.

Durante la especificación literal de la fecha, es deseable indicar el año, el mes y el día. Si no, el compilador mostrará un aviso sobre la entrada literal no completa.

Ejemplos:

```
datetime NY=D'2015.01.01 00:00';      // hora de inicio del año 2015
datetime d1=D'1980.07.19 12:30:27';   // año mes día horas minutos segundos
datetime d2=D'19.07.1980 12:30:27';   // igual a D'1980.07.19 12:30:27';
datetime d3=D'19.07.1980 12';         // igual a D'1980.07.19 12:00:00'
datetime d4=D'01.01.2004';            // igual a D'01.01.2004 00:00:00'
datetime compilation_date=__DATE__;              // fecha de compilación
datetime compilation_date_time=__DATETIME__;     // fecha y hora de compilación
datetime compilation_time=__DATETIME__-__DATE__;// hora de compilación
//--- ejemplos de declaraciones que provocarán los avisos del compilador
datetime warning1=D'12:30:27';        // igual a D'[fecha de compilación] 12:30:27'
datetime warning2=D'';                // igual a __DATETIME__
```

Véase también

Estructura de fecha, Fecha y hora, TimeToString, StringToTime

© 2000-2025, MetaQuotes Ltd.

Tipo color

El tipo color sirve para almacenar la información sobre el color y ocupa en la memoria 4 bytes. El primer byte no se cuenta, los demás 3 bytes contienen los componentes RGB.

Las constantes de colores pueden ser representadas de tres formas distintas: de forma literal, con números enteros o mediante un nombre (sólo para los colores-Web concretos).

La representación literal se compone de tres partes que representan valores numéricos de la intensidad de tres componentes principales: rojo (red), verde (green), azul (blue). La constante se mete entre comillas simples y se empieza con el signo C. Los valores numéricos de la intensidad del componente de color se encuentra entre 0 y 255.

La representación con números enteros se realiza a través de un número decimal o hexadecimal. El número hexadecimal tiene la siguiente forma 0x00BBGGRR, donde RR es la intensidad del color rojo, GG - verde y BB - azul. Las constantes decimales no tienen la expresión directa en RGB. Estas representan el valor decimal de la expresión hexadecimal de números enteros.

Los colores concretos expresan un conjunto de colores-Web.

Ejemplos:

```
//--- literales
C'128,128,128'    // gris
C'0x00,0x00,0xFF' // azul
//nombres de colores
clrRed               // rojo
clrYellow            // amarillo
clrBlack             // negro
//--- representaciones con números enteros
0xFFFFFF          // blanco
16777215          // blanco
0x008000          // verde
32768             // verde
```

Véase también

Colores Web, ColorToString, StringToColor, Conversión de tipos

© 2000-2025, MetaQuotes Ltd.

Tipo bool

El tipo bool sirve para almacenar los valores lógicos true (verdadero) o false (falso) la representacion de los cuales es 1 o 0 respectivamente.

Ejemplos:

```
bool a = true;
bool b = false;
bool c = 1;
```

La representación interna es un número entero de tamano de 1 byte. Cabe señalar que en las expresiones lógicas se puede utilizar en vez del tipo bool también otros tipos enteros o reales o representaciones de estos tipos, el compilador igual no va a señalar ningun error. En este caso el valor cero va a ser interpretado como false y los demás como true.

Ejemplos:

```
   int i=5;
   double d=-2.5;
   if(i) Print("i = ",i," y tiene el valor true");
   else Print("i = ",i,"y tiene el valor false");

   if(d) Print("d = ",d," y tiene el valor true");
   else Print("d = ",d," y tiene el valor false");

   i=0;
   if(i) Print("i = ",i," y tiene el valor true");
   else Print("i = ",i," y tiene el valor false");

   d=0.0;
   if(d) Print("d = ",d," y tiene el valor true");
   else Print("d = ",d," y tiene el valor false");

//--- resultados de ejecución
//   i= 5 y tiene el valor true
//   d= -2.5 y tiene el valor true
//   i= 0 y tiene el valor false
//   d= 0 y tiene el valor false
```

Véase también

Operaciones lógicas, Prioridades y orden de las operaciones

© 2000-2025, MetaQuotes Ltd.

Enumeraciones

Los datos del tipo enumerativo enum se refieren a una cantidad limitada de datos. La definición del tipo enumerativo:

```
enum nombre del tipo enumerativo
  {
   lista de valores
  };
```

La lista de valores es una lista de variables separadas por comas.

Ejemplo:

```
enum months  // enumeración de las constantes concretas
   {
    January,
    February,
    March,
    April,
    May,
    June,
    July,
    August,
    September,
    October,
    November,
    December
   };
```

Después de declarar una enumeración aparece un nuevo tipo de datos de números enteros de 4 bytes. La declaración del nuevo tipo de datos permite al compilador controlar de una forma estricta los parámetros transferidos porque la enumeración establece nuevas constantes concretas. En el ejemplo de arriba la constante concreta January tiene el valor 0, February tiene el valor 1, December tiene el valor 11.

Regla: si una constante concreta, siendo un elemento de enumeración, no tiene adjudicado un valor concreto entonces su valor se forma de una manera automática. Si se trata del primer elemento de enumeración se le va a adjudicar el valor 0. Para todos los elementos posteriores los valores se calculan a base del valor del elemento anterior sumándole un uno.

Ejemplo:

```
enum intervals  // enumeración de las constantes concretas
   {
    month=1,     // intervalo de un mes
    two_months,  // dos meses
    quarter,     // tres meses - trimestre
    halfyear=6,  // semestre
    year=12,     // año - 12 meses
   };
```

© 2000-2025, MetaQuotes Ltd.

Observaciones

- A diferencia de C++ el tamaño interno de la representación de un tipo enumerativo en MQL5 siempre es de 4 bytes. Es decir, sizeof(months) devolverá el valor 4.
- A diferencia de C++ en MQL5 no se puede declarar una enumeración anónima. Es decir, después de la palabra clave enum siempre hay que indicar un nombre único.

Véase también

Conversión de tipos

© 2000-2025, MetaQuotes Ltd.

Tipos reales (double, float)

Los tipos reales (o tipos de punto flotante) representan valores que contienen la parte fraccionaria. En el lenguaje MQL5 existen dos tipos para los números con punto flotante. El modo de representar los números reales en la memoria del ordenador se rige por el estándar IEEE 754 y no depende de las plataformas, sistemas operativos y lenguajes de programación.

Tipo	Tamaño en bytes	Valor mínimo positivo	Valor máximo	Análogo en el lenguaje C++
float	4	1.175494351e-38	3.402823466e+38	float
double	8	2.225073858507 2014e-308	1.797693134862 3158e+308	double

double

El tipo de número real double ocupa 64 bits (1 bit de signo, 11 bits de exponente y 52 bits de mantisa).

float

El tipo de número real float ocupa 32 bits (1 bit de signo, 8 bits de exponente y 23 bits de mantisa).

vector

Array numérico unidimensional del tipo double. La memoria para los datos se asigna dinámicamente. Las propiedades del vector se pueden recuperar usando métodos, y el tamaño del vector se puede cambiar. En las funciones de plantilla se puede utilizar la notación vector<double>.

vectorf

Array numérico unidimensional del tipo float, puede usarse en lugar de vector, si la pérdida de precisión no tiene importancia. En las funciones de plantilla se puede utilizar la notación vector<float>.

vectorc

Array numérico unidimensional del tipo complex, está diseñado para trabajar con números complejos. En las funciones de plantilla se puede utilizar la notación vector<complex>. Las operaciones sobre vectores de tipo vectorc aún no están implementadas.

matrix

Una matriz es un array numérico bidimensional del tipo double. La memoria para los elementos de la matriz se asigna dinámicamente. Las propiedades de la matriz se pueden obtener usando métodos, y

© 2000-2025, MetaQuotes Ltd.

el tamaño del vector se puede cambiar. En las funciones de plantilla se puede utilizar la notación matrix<double>.

matrixf

Array numérico bidimensional del tipo float, puede utilizarse en lugar de matrix si la pérdida de precisión no tiene importancia. En las funciones de plantilla se puede utilizar la entrada matrix<float>.

matrixc

Array numérico bidimensional del tipo complex, está diseñado para trabajar con números complejos. En las funciones de plantilla se puede utilizar la notación matrix<complex>. En estos momentos, las operaciones sobre matrices del tipo matrixc aún no están implementadas.

El nombre double significa que la precisión de estos números es dos veces más que la precisión de los números del tipo float. En mayoría de los casos el tipo double es más cómodo. En muchos casos la precisión limitada de los números float simplemente es insuficiente. La razón de utilizar todavía el tipo float se encuentra en el ahorro de la memoria durante el almacenamiento (es importante para las grandes cantidades de matrices de números reales).

Las constantes de punto flotante se componen de la parte entera, punto (.) y parte fraccionaria. La parte entera y fraccionaria es una sucesión de números decimales.

Ejemplos:

```
double a=12.111;
double b=-956.1007;
float  c =0.0001;
float  d =16;
```

Existe el modo científico de escribir las constantes reales. A menudo esta modo de escribir es más compacto en comparación con la forma tradicional.

Ejemplo:

```
double c1=1.12123515e-25;
double c2=0.000000000000000000000000112123515; // 24 ceros despues del punto decima

Print("1. c1 = ",DoubleToString(c1,16));
// Resultado: 1. c1 = 0.0000000000000000

Print("2. c1 = ",DoubleToString(c1,-16));
// Resultado: 2. c1 = 1.1212351499999999e-025

Print("3. c2 = ",DoubleToString(c2,-16));
// Resultado: 3. c2 = 1.1212351499999999e-025
```

Hay que recordar que los números reales se almacenan en la memoria del ordenador con una cierta precisión limitada en el sistema binario, mientras que el sistema decimal es de uso general. Por eso

© 2000-2025, MetaQuotes Ltd.

muchos números que se escriben en el sistema decimal, en el sistema binario pueden ser escritos sólo en forma de fracción contínua.

Por ejemplo, en el ordenador los números 0.3 y 0.7 están representados por las fracciones contínuas, mientras que el número 0.25 se guarda de forma exacta porque es la potencia de 2.

Por esta razón no se recomienda de ninguna manera comparar la igualdad de dos números reales porque esta comparación no es correcta.

Ejemplo:

```
void OnStart()
  {
//---
   double three=3.0;
   double x,y,z;
   x=1/three;
   y=4/three;
   z=5/three;
   if(x+y==z)
      Print("1/3 + 4/3 == 5/3");
   else
      Print("1/3 + 4/3 != 5/3");
// Resultado: 1/3 + 4/3 != 5/3
  }
```

Si en alguna ocasión es necesario comparar la igualdad de dos números reales, entonces se puede hacerlo de dos maneras distintas. El primer modo consiste en la comparación de diferencia entre dos números con un valor pequeño que marca la precisión de comparación.

Ejemplo:

```
bool EqualDoubles(double d1,double d2,double epsilon)
  {
   if(epsilon<0)
      epsilon=-epsilon;
//---
   if(d1-d2>epsilon)
      return false;
   if(d1-d2<-epsilon)
      return false;
//---
   return true;
  }
void OnStart()
  {
   double d_val=0.7;
   float  f_val=0.7;
   if(EqualDoubles(d_val,f_val,0.000000000000001))
      Print(d_val,"equals",f_val);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("Different: d_val = ",DoubleToString(d_val,16),"  f_val = ",DoubleToStrinc
// Resultado: Different: d_val= 0.7000000000000000   f_val= 0.6999999880790710
  }
```

Cabe mencionar que el valor del parámetro epsilon en el ejemplo de arriba, no puede ser menos que la constante predeterminada DBL_EPSILON. El valor de esta constante es 2.2204460492503131e-016. Para el tipo float la constante correspondiente es FLT_EPSILON = 1.192092896e-07. El sentido de estos valores es siguiente: se trata del valor mínimo que satisface la condición 1.0+DBL_EPSILON != 1.0 (para los números del tipo float 1.0+FLT_EPSILON != 1.0).

El segundo modo supone la comparación de la diferencia normalizada de dos números reales con el valor cero. Es inutil comparar la diferencia de los números normalizados con el cero porque el resultado de cualquier operación matemática con los números normalizados no va a ser normalizado.

Ejemplo:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0)
      return(true);
   else
      return(false);
  }
void OnStart()
  {
   double d_val=0.3;
   float  f_val=0.3;
   if(CompareDoubles(d_val,f_val))
      Print(d_val,"equals",f_val);
   else
      Print("Different: d_val=",DoubleToString(d_val,16),"  f_val=",DoubleToString(f_v
// Resultado: Different: d_val= 0.3000000000000000   f_val= 0.3000000119209290
  }
```

Como resultado de algunas operaciones matemáticas del coprocesador se puede obtener un número real extendido, el que no se puede utilizar en las operaciones matemáticas y operaciones de comparación porque el resultado de ejecución de las operaciones con números reales extendidos no está definido. Por ejemplo, tratando de calcular el arcseno de 2 se obtiene el infinito negativo.

Ejemplo:

```
   double abnormal = MathArcsin(2.0);
   Print("MathArcsin(2.0) =",abnormal);
// Resultado:  MathArcsin(2.0) = -1.#IND
```

A parte del infinito negativo existe el infinito positivo y NaN (no es un número). Para determinar que el número es extendido se puede utilizar la función MathIsValidNumber(). Según el estándar IEEE ellos tienen una representación informática especial. Por ejemplo, el infinito positivo para el tipo double tiene la representación de bit 0x7FF0 0000 0000 0000.

Ejemplos:

```
struct str1
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   double d;
  };
struct str2
  {
   long l;
  };

//--- empecemos
   str1 s1;
   str2 s2;
//---
   s1.d=MathArcsin(2.0);         // obtenemos número extendido -1.#IND
   s2=s1;
   printf("1.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFFF000000000000;      // número extendido -1.#QNAN
   s1=s2;
   printf("2.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF7000000000000;      // nonúmero máximo SNaN
   s1=s2;
   printf("3.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF8000000000000;      // nonúmero mínimo QNaN
   s1=s2;
   printf("4.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FFF000000000000;      // nonúmero máximo QNaN
   s1=s2;
   printf("5.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF0000000000000;      // infinito positivo 1.#INF y nonúmero mínimo SNaN
   s1=s2;
   printf("6.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFF0000000000000;      // infinito negativo -1.#INF
   s1=s2;
   printf("7.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x8000000000000000;      // cero negativo -0.0
   s1=s2;
   printf("8.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FE0000000000000;      // 0.5
   s1=s2;
   printf("9.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FF0000000000000;      // 1.0
```

© 2000-2025, MetaQuotes Ltd.

```
   s1=s2;
   printf("10.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FEFFFFFFFFFFFFF;     // número máximo normalizado (MAX_DBL)
   s1=s2;
   printf("11.  %.16e %I64X",s1.d,s2.l);
//---
   s2.l=0x0010000000000000;     // mínimo normalizado positivo (MIN_DBL)
   s1=s2;
   printf("12.  %.16e %.16I64X",s1.d,s2.l);
//---
   s1.d=0.7;                    // indicamos que el número 0.7 - es una fracción conti
   s2=s1;
   printf("13.  %.16e %.16I64X",s1.d,s2.l);
/*
1.  -1.#IND00 FFF8000000000000
2.  -1.#QNAN0 FFFF000000000000
3.   1.#SNAN0 7FF7000000000000
4.   1.#QNAN0 7FF8000000000000
5.   1.#QNAN0 7FFF000000000000
6.   1.#INF00 7FF0000000000000
7.  -1.#INF00 FFF0000000000000
8.  -0.000000 8000000000000000
9.   0.500000 3FE0000000000000
10.  1.000000 3FF0000000000000
11.  1.7976931348623157e+308 7FEFFFFFFFFFFFFF
12.  2.2250738585072014e-308 0010000000000000
13.  6.9999999999999996e-001 3FE6666666666666
*/
```

Véase también

DoubleToString, NormalizeDouble, Constantes de tipos numéricos

© 2000-2025, MetaQuotes Ltd.

Números complejos (complex)

El tipo complex incorporado es una estructura con dos campos double:

```
struct complex
  {
   double            real;    // parte real
   double            imag;    // parte imaginaria
  };
```

El tipo "complex" puede transmitirse mediante un valor en calidad de parámetro para las funciones MQL5 (a diferencia de las estructuras normales, que se transmiten mediante un enlace). Para las funciones importadas desde una DLL, el tipo "complex" debe transmitirse solo mediante un enlace.

Para describir las constantes complejas, se usa el sufijo 'i':

```
complex square(complex c)
  {
   return(c*c);
  }
void OnStart()
  {
   Print(square(1+2i));  // se transmite una constante como parámetro
  }
// se mostrará "(-3,4)", la representación de línea del número complejo
```

Para los números complejos, en estos momentos están disponibles solo las operaciones sencillas: =, +, -, *, /, +=, -=, *=, /=, ==, !=.

En el futuro, se añadirán funciones matemáticas adicionales: obtención del valor absoluto, seno, coseno y muchas otras.

vectorc

Array numérico unidimensional del tipo complex, está diseñado para trabajar con números complejos. En las funciones de plantilla se puede utilizar la notación vector<complex>. Las operaciones sobre vectores de tipo vectorc aún no están implementadas.

matrixc

Array numérico bidimensional del tipo complex, está diseñado para trabajar con números complejos. En las funciones de plantilla se puede utilizar la notación matrix<complex>. En estos momentos, las operaciones sobre matrices del tipo matrixc aún no están implementadas.

© 2000-2025, MetaQuotes Ltd.

Tipo string

El tipo string sirve para guardar las cadenas de caracteres. Una cadena de caracteres es una sucesión de caracteres en formato Unicode con el cero al final. A una variable string se le puede asignar una constante literal. Una constante literal es una sucesión de caracteres Unicode encerada entre comillas dobles: "Es una constante literal".

Para poder introducir una comilla doble (") dentro de la cadena hay que interponerle el signo de barra inversa (\). Cualquier constante de signo especial, si le interviene el signo de barra inversa (\), puede ser introducida en la cadena.

Ejemplos:

```
string svar="This is a character string";
string svar2=StringSubstr(svar,0,4);
Print("Símbolo de copyright\t\x00A9");
FileWrite(handle,"esta cadena contiene el símbolo de avance de línea \n");
string MT5path="C:\\Program Files\\MetaTrader 5";
```

Para hacer el código fuente más cómodo para leer, las cadenas constantes largas se puede dividir en partes sin la operación de adición. Durante el proceso de compilación, todas estas partes se unirán en una cadena larga:

```
//--- declararemos una cadena constante larga
   string HTML_head="<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"'
                    " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n"
                    "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n"
                    "<head>\n"
                    "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ut
                    "<title>Trade Operations Report</title>\n"
                    "</head>";
//--- mostraremos la cadena constante en el diario
   Print(HTML_head);
  }
```

Métodos incorporados del tipo string

Para trabajar con cadenas, podemos usar las funciones de cadena, las funciones de conversión y los métodos incorporados del tipo string que mostramos en el recuadro:

Método string	Análogo	Descripción
Constructor string(const int len)		Construye una cadena de la longitud indicada
string[] tanto para la lectura, como para la escritura. El índice debe		Proporciona acceso al elemento de cadena según el índice especificado

© 2000-2025, MetaQuotes Ltd.

Método string	Análogo	Descripción
encontrarse en los límites de BufferSize()		
static string string.Init(const int len, const ushort character);	StringInit	Inicializa una cadena con los símbolos y el tamaño indicados
void string.Fill(const ushort character);	StringFill	Rellena una cadena con el símbolo indicado
int string.Len();	StringLen	Retorna el número de caracteres en una cadena
int string.BufferSize();	StringBufferLen	Retorna el tamaño de un búfer distribuido para una cadena
bool string.SetLen(const int new_len);	StringSetLength	Establece para una cadena la longitud indicada en caracteres
bool string.Reserve(const int buffer_len);	StringReserve	Reserva en la memoria para una cadena un búfer del tamaño indicado
bool string.Add(const string substring);	StringAdd	Añade una subcadena especificada al final
int string.Concatenate(const scalar val1, const scalar val2...);	StringConcatenate	Forma una cadena que consta de los parámetros transmitidos
array string.Split(const ushort separator, const bool long_separator);	StringSplit	Retorna una matriz de cadenas según el separador indicado
int string.Compare(const string str, const bool case_sensivity);	StringCompare	Realiza una comparación con la cadena especificada y retorna 1 si la primera cadena es mayor que la segunda; 0 - si las cadenas son iguales; -1 (menos uno) - si la primera cadena es menor que la segunda
string string.Substr(const int start_pos, const int len);	StringSubstr	Extrae una subcadena de la posición indicada
int string.Find(const string substr, const int pos);	StringFind	Retorna el índice de la posición desde la que comienza la subcadena necesaria
void string.ToLower();	StringToLower	Pasa todos los caracteres a minúscula
void string.ToUpper();	StringToUpper	Pasa todos los caracteres a mayúscula
int string.TrimLeft();	StringTrimLeft	Elimina los espacios, así como el movimiento del carro y los

© 2000-2025, MetaQuotes Ltd.

Método string	Análogo	Descripción
		caracteres de tabulación a la izquierda.
int string.TrimRight()	StringTrimRight	Elimina los espacios, así como el movimiento del carro y los caracteres de tabulación a la derecha.
void string.Double(const double var, const int digits=8);	DoubleToString	Convierte una cadena en un número del tipo double
void string.Enum(const enum value);	EnumToString	Convierte un valor de enumeración de cualquier tipo en una cadena
void string.Integer(const int value, const int str_len=0, const ushort fill=' ');	IntegerToString	Convierte una cadena en un número del tipo long
void string.CharArray(const uchar array[], const int start_pos=0, const int len=-1, const uint cp=CP_ACP);	CharArrayToString	Convierte parte de una matriz del tipo uchar en una cadena
void string.ShortArray(const ushort array[], const int start_pos=0, const int len=-1);	ShortArrayToString	Copia parte de una matriz del tipo ushort en una cadena
void string.Time(const datetime dt,const int mode=TIME_DATE|TIME_MINUTES);	TimeToString	Convierte un valor datetime en una cadena del formato "yyyy.mm.dd hh:mi".
void string.Format(const string format_str);	StringFormat	Formatea los parámetros obtenidos en una cadena

Véase también

Conversión de tipos, Funciones de cadenas de caracteres, FileOpen, FileReadString, FileWriteString

© 2000-2025, MetaQuotes Ltd.

Estructuras, clases e interfaces

Estructuras

Una estructura es un conjunto de elementos del tipo libre, salvo el tipo void. De esa manera, la estructura une los datos de diferentes tipos que están vinculados de forma lógica.

Declaración de estructura

El tipo de datos estructural se define de la siguiente manera:

```
struct nombre_de_estructura
  {
   descripción_de_elementos
  };
```

No se puede usar el nombre de la estructura en calidad del identificador (nombre de la variable o función). Hay que tener en cuenta que en MQL5 los elementos de una estructura siguen directamente uno detrás del otro sin que sean alineados. En el lenguaje C++ este comando se proporciona al compilador mediante la instrucción

```
#pragma pack(1)
```

Si hace falta hacer otra alineación dentro de la estructura, es necesario utilizar los elementos "de relleno" adicionales de tamaño necesario.

Ejemplo:

```
struct trade_settings
  {
   uchar  slippage;     // valor del deslizamiento permitido - tamaño 1 byte
   char   reserved1;    // 1 byte de permiso
   short  reserved2;    // 2 bytes de permiso
   int    reserved4;    // otros 4 bytes de permiso. Aseguramos la alineación al marge
   double take;         // valor del precio de fijación del beneficio
   double stop;         // valor del precio del stop de protección
  };
```

Esta descripción de alineación de las estructuras es necesaria únicamente para la transmisión a las funciones dll importadas.

Atención: este ejemplo refleja los datos proyectados de una manera errónea. Sería mejor declarar al principio los datos take y stop de mayor tamaño que el tipo double, y luego declarar el elemento slippage del tipo uchar. En este caso, la presentación interna de los datos siempre va a ser igual independientemente del valor indicado en #pragma pack().

Si la estructura contiene las variables del tipo string y/o el objeto del array dinámico, entonces para esta estructura el compilador asigna un constructor implícito donde se efectúa la anulación de todos los elementos del tipo string y la inicialización correcta para el objeto del array dinámico.

Estructuras simples

© 2000-2025, MetaQuotes Ltd.

Las estructuras que no contengan cadenas, objetos de clase, punteros y objetos de array dinámico se llaman estructuras simples. Las variables de las estructuras sencillas, así como sus arrays se pueden transmitir como parámetros a las funciones importadas de DLL.

El copiado de estructuras sencillas está permitido solo en dos casos:

- si los objetos pertenecen a un tipo de estructura
- si los objetos están relacionados entre sí por una línea de herencia, es decir, una estructura es heredera de la otra.

Mostraremos esto con ejemplos y crearemos la estructura de usuario CustomMqlTick, idéntica en su composición a la estructura incorporada MqlTick. El compilador no permitirá intentos de copiar el valor del objeto MqlTick en un objeto del tipo CustomMqlTick. La conversión directa al tipo necesario también provocará el error de compilación:

```
      //--- copiar estructuras simples de tipos diferentes está prohibido
      my_tick1=last_tick;               // el compilador aquí dará error

      //--- tampoco está permitido convertir estructuras de diferente tipo
      my_tick1=(CustomMqlTick)last_tick;// el compilador aquí dará error
```

Por eso, solo queda una opción: copiar los valores de los miembros de la estructura elemento a elemento. Pero, en este caso, está permitido copiar los valores de los objetos de un mismo tipo CustomMqlTick.

```
      CustomMqlTick my_tick1,my_tick2;
      //--- así, sí está permitido copiar objetos de una misma estructura CustomMqlTic
      my_tick2=my_tick1;

      //--- vamos a crear un array de objetos de la estructura simple CustomMqlTick y
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
```

Como comprobación, se llama la función ArrayPrint() para mostrar el diario de cambios del array *arr[]*.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- creamos una estructura igual que la MqlTick incorporada
   struct CustomMqlTick
     {
      datetime          time;          // Hora de la última actualización de precio
      double            bid;           // Precio Bid actual
      double            ask;           // Precio Ask actual
      double            last;          // Precio actual de la última transacción (Last
      ulong             volume;        // Volumen actual para el último precio Last
      long              time_msc;      // Hora de la última actualización en milisegu
      uint              flags;         // Banderas de los ticks
     };
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- obtenemos los valores del último tick
   MqlTick last_tick;
   CustomMqlTick my_tick1,my_tick2;
//--- tratamos de copiar los datos de MqlTick a CustomMqlTick
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      //--- está prohibido copiar estructuras simples no relacionadas
      //1. my_tick1=last_tick;               // el compilador aquí dará error

      //--- tampoco está permitido convertir estructuras no relacionadas entre sí
      //2. my_tick1=(CustomMqlTick)last_tick;// el compilador aquí dará error

      //--- por eso, copiamos los miembros de la estructura elemento a elemento
      my_tick1.time=last_tick.time;
      my_tick1.bid=last_tick.bid;
      my_tick1.ask=last_tick.ask;
      my_tick1.volume=last_tick.volume;
      my_tick1.time_msc=last_tick.time_msc;
      my_tick1.flags=last_tick.flags;

      //--- así, sí está permitido copiar objetos de una misma estructura CustomMqlTic
      my_tick2=my_tick1;

      //--- vamos a crear un array de objetos de la estructura simple CustomMqlTick y
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
      ArrayPrint(arr);
//--- ejemplo de muestra de los valores de un array que contiene objetos del tipo Cust
      /*
                         [time]    [bid]    [ask]    [last] [volume]     [time_msc] [flags]
      [0] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157       2
      [1] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157       2
      */
     }
   else
      Print("SymbolInfoTick() failed, error = ",GetLastError());
  }
```

El segundo ejemplo muestra las posibilidades de copiado de estructuras simples según la línea de herencia. Tenemos la estructura básica Animal, de la cual se generan mediante herencia las estructuras Cat y Dog. Podemos copiar entre sí los objetos Animal y Cat, Animal y Dog, pero no podemos copiar entre sí Cat y Dog, aunque ambos sean descendientes de la estructura Animal.

```
//--- estructura para la descripción de perros
struct Dog: Animal
  {
   bool              hunting;       // raza cazadora
  };
```

© 2000-2025, MetaQuotes Ltd.

```
//--- estructura para la descripción de gatos
struct Cat: Animal
  {
   bool              home;          // raza doméstica
  };
//--- creamos los objetos de las subclases
   Dog dog;
   Cat cat;
//--- podemos copiar del padre al descendiente (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // los perros saben nadar
//--- no se pueden copiar objetos de las subestructuras (Dog != Cat)
   cat=dog;        // el compilador aquí dará error
```

Código completo del ejemplo:

```
//--- estructura básica para la descripción de animales
struct Animal
  {
   int               head;          // número de cabezas
   int               legs;          // número de patas
   int               wings;         // número de alas
   bool              tail;          // existencia de cola
   bool              fly;           // vuela
   bool              swim;          // nada
   bool              run;           // corre
  };
//--- estructura para la descripción de perros
struct Dog: Animal
  {
   bool              hunting;       // raza cazadora
  };
//--- estructura para la descripción de gatos
struct Cat: Animal
  {
   bool              home;          // raza doméstica
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- creamos un objeto del tipo básico Animal y lo describimos
   Animal some_animal;
   some_animal.head=1;
   some_animal.legs=4;
   some_animal.wings=0;
   some_animal.tail=true;
   some_animal.fly=false;
```

© 2000-2025, MetaQuotes Ltd.

```
   some_animal.swim=false;
   some_animal.run=true;
//--- creamos objetos de los subtipos
   Dog dog;
   Cat cat;
//--- podemos copiar del padre al descendiente (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // los perros saben nadar
//--- no se pueden copiar objetos de las subestructuras (Dog != Cat)
   //cat=dog;        // el compilador aquí dará error
//--- por eso solo se puede copiar elemento a elemento
   cat.head=dog.head;
   cat.legs=dog.legs;
   cat.wings=dog.wings;
   cat.tail=dog.tail;
   cat.fly=dog.fly;
   cat.swim=false;   // los gatos no saben nadar
//--- se pueden copiar los valores del padre al descendiente
   Animal elephant;
   elephant=cat;
   elephant.run=false;// los elefantes no pueden correr
   elephant.swim=true;// los elefantes nadan
//--- creamos un array
   Animal animals[4];
   animals[0]=some_animal;
   animals[1]=dog;
   animals[2]=cat;
   animals[3]=elephant;
//--- realizamos la impresión
   ArrayPrint(animals);
//--- resultado de la ejecución
/*
        [head] [legs] [wings] [tail] [fly] [swim] [run]
   [0]      1      4        0   true false  false  true
   [1]      1      4        0   true false   true  true
   [2]      1      4        0   true false  false false
   [3]      1      4        0   true false   true false
*/
  }
```

Otro método para copiar tipos simples es el uso de las uniones, para ello, los objetos de estas estructuras deberán ser miembros de una misma unión, vea los ejemplos en union.

Acceso a los elementos de la estructura

El nombre de la estructura es un tipo de datos nuevo y permite declarar las variables de este tipo. Se puede declarar la estructura sólo una vez dentro de un proyecto. El acceso a los elementos de las estructuras se realiza mediante operación punto (.).

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
struct trade_settings
  {
   double take;         // valor del precio de fijación del beneficio
   double stop;         // valor del precio del stop de protección
   uchar  slippage;     // valor del deslizamiento permitido
  };
//--- creamos e inicializamos la variable del tipo trade_settings
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

pack para alinear los campos de estructuras y las clases

El atributo especial pack permite establecer la alineación de los campos de una estructura o clase.

```
pack([n])
```

donde n - es uno de los valores siguientes 1,2,4,8 o 16. Puede no existir.

Ejemplo:

```
   struct pack(sizeof(long)) MyStruct
     {
      // los miembros de la estructura se alinearán en el borde del byte 8
     };
o
   struct MyStruct pack(sizeof(long))
     {
      // los miembros de la estructura se alinearán en el borde del byte 8
     };
```

Por defecto, para las estructuras se usa pack(1). Esto significa que en la memoria los miembros de la estructura se ubican uno tras otro, y que el tamaño de la estructura es igual a la suma de los tamaños de sus miembros.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- estructura simple sin alineación
   struct Simple_Structure
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- declaramos el ejemplar de la estructura simple
```

© 2000-2025, MetaQuotes Ltd.

```
   Simple_Structure s;
//--- mostramos el tamaño de cada miembro de la estructura
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- nos aseguramos de que el tamaño de la estructura POD sea igual a la suma de los
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  Resultado:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=15
*/
  }
```

La alineación de los campos de una estructura podría ser necesaria al intercambiar datos con terceras bibliotecas (*.DLL), en las que dicha alineación es necesaria.

Vamos a mostrar con ejemplos cómo funciona la alineación. Tomemos una estructura de cuatro miembros sin alinear.

```
//--- estructura simple sin alineación
   struct Simple_Structure pack() // el tamaño no se indica, habrá alineación por el t
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
//--- declaramos un ejemplar de estructura simple
   Simple_Structure s;
```

Los campos de la estructura se ubican en la memoria uno tras otro, de acuerdo con el orden y el tamaño del tipo. El tamaño de la estructura es igual a 15, el desplazamiento hacia los campos de la estructura en las matrices será indefinido.



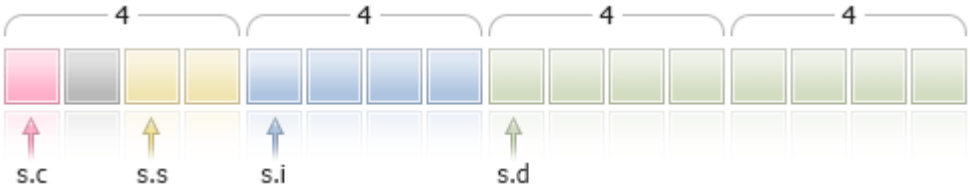
Declaramos ahora esta misma estructura con una alineación de 4 bytes e iniciamos el código.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- estructura sencilla con alineación de 4 bytes
   struct Simple_Structure pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- declaramos el ejemplar de la estructura simple
   Simple_Structure s;
//--- mostramos el tamaño de cada miembro de la estructura
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- nos aseguramos de que el tamaño de la estructura POD no sea ahora igual a la sur
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  Resultado:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=16 // el tamaño de la estructura ha cambiado
*/
  }
```

El tamaño de la estructura ha cambiado de tal forma que todos los miembros con un tamaño de 4 bytes o más tengan un desplazamiento respecto al inicio de la estructura que sea múltiplo de 4 bytes. Los miembros de menor tamaño se alinearán hacia el borde de su tamaño (por ejemplo, 2 para short). Aquí vemos el aspecto que tiene esto, el byte añadido se muestra en color gris.



En este caso, tras el miembro **s.c** se ha añadido 1 byte, para que el cmapo **s.s** (sizeof(short)==2) tenga un borde de 2 bytes (alineación para el tipo short).

El desplazamiento hacia el comienzo de la estructura también será alineado en el borde de 4 bytes, es decir, para Simple_Structure arr[], las direcciones de los elementos a[0], a[1], a[n] serán múltiplos de 4 bytes.

Vamos a ver otras dos estructuras que constan de tipos iguales con una igualación de 4 bytes, pero en este caso, además, el orden secuencial de los miembros será diferente. En la primera estructura, los miembros se ubican en orden ascendente según el tamaño del tipo.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- estructura simple con alineación en el borde de 4 bytes
   struct CharShortInt pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(double)=4
     };
//--- declaramos un ejemplar de estructura simple
   CharShortInt ch_sh_in;
//--- mostramos el tamaño de cada miembro de la estructura
   Print("sizeof(ch_sh_in.c)=",sizeof(ch_sh_in.c));
   Print("sizeof(ch_sh_in.s)=",sizeof(ch_sh_in.s));
   Print("sizeof(ch_sh_in.i)=",sizeof(ch_sh_in.i));

//--- nos aseguramos de que el tamaño de la estructura POD sea igual a la suma de los
   Print("sizeof(CharShortInt)=",sizeof(CharShortInt));
/*
  Resultado:
   sizeof(ch_sh_in.c)=1
   sizeof(ch_sh_in.s)=2
   sizeof(ch_sh_in.i)=4
   sizeof(CharShortInt)=8
*/
  }
```

Como podemos ver, el tamaño de la estructura es igual a 8, y consta de dos bloques de 4 bytes. En el primer bloque se ubican los campos con los tipos char y short; en el segundo, el campo con el tipo int.



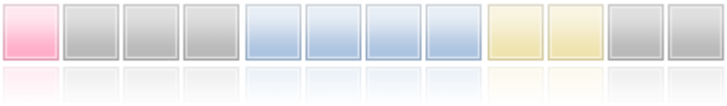
Ahora, a partir de la primera estructura, creamos la segunda, que se distingue solo en el orden secuencial de los campos. Colocamos el miembro del tipo short al final.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- estructura simple con alineación en el borde de 4 bytes
   struct CharIntShort pack(4)
     {
      char              c; // sizeof(char)=1
      int               i; // sizeof(double)=4
      short             s; // sizeof(short)=2
     };
```

© 2000-2025, MetaQuotes Ltd.

```
//--- declaramos un ejemplar de estructura simple
   CharIntShort ch_in_sh;
//--- mostramos el tamaño de cada miembro de la estructura
   Print("sizeof(ch_in_sh.c)=",sizeof(ch_in_sh.c));
   Print("sizeof(ch_in_sh.i)=",sizeof(ch_in_sh.i));
   Print("sizeof(ch_in_sh.s)=",sizeof(ch_in_sh.s));
//--- nos aseguramos de que el tamaño de la estructura POD sea igual a la suma de los
   Print("sizeof(CharIntShort)=",sizeof(CharIntShort));
/*
  Resultado:
   sizeof(ch_in_sh.c)=1
   sizeof(ch_in_sh.i)=4
   sizeof(ch_in_sh.s)=2
   sizeof(CharIntShort)=12
*/
  }
```

Aunque la propia composición de la estructura no ha variado, el cambio de orden de los miembros ha provocado un aumento del tamaño de la propia estructura.



Al heredar, también es necesario tener en cuenta la alineación. Vamos a mostrar un ejemplo de estructura sencilla Parent, que tiene un miembro del tipo char. El tamaño de esta estructura sin alineación es igual a 1.

```
   struct Parent
     {
      char              c;     // sizeof(char)=1
     };
```

Creamos la clase hija Children añadiendo un miembro del tipo short (sizeof(short)=2).

```
   struct Children pack(2) : Parent
     {
      short             s;    // sizeof(short)=2
     };
```

Como resultado, al realizar una alineación en 2 bytes, el tamaño de la estructura será igual a 4, aunque el tamaño de los propios miembros en ella sea igual a 3. En este ejemplo, a la clase padre Parent se le asignarán 2 bytes para que el acceso al campo short de la clase hija sea igual a 2 bytes.

Saber cómo se distribuye la memoria de los miembros de la estructura es imprescindible si un programa MQL5 interactúa con datos ajenos mediante registro/lectura al nivel de archivos o flujos.

En la Biblioteca Estándar, en el catálogo MQL5\Include\WinAPI se muestran las funciones para trabajar con las funciones WinAPI. Dichas funciones usan estructuras con alineaciones establecidas para aquellos casos en los que este hecho es necesario para trabajar con WinAPI.

© 2000-2025, MetaQuotes Ltd.

offsetof - es un comanado especial directamente relacionado con el atributo pack. Este permite obtener el desplazamiento del miembro respecto al inicio de la estructura.

```
//--- declaramos la variable de tipo Children
   Children child;
//--- averiguamos el desplazamiento respecto al comienzo de la estructura
   Print("offsetof(Children,c)=",offsetof(Children,c));
   Print("offsetof(Children,s)=",offsetof(Children,s));
/*
  Resultado:
   offsetof(Children,c)=0
   offsetof(child.ы)=2
*/
```

Especificador final

La existencia del especificador final al declarar la estructura, prohíbe la posterior herencia a partir de ella. Si la estructura es tal que no haya necesidad de introducir cambios posteriormente, o los cambios no están permitidos por motivos de seguridad, declárela con el especificador final. Además, todos los miembros de la estructura también se considerarán implícitamente como "final".

```
struct settings final
  {
  //--- cuerpo de la estructura
  };

struct trade_settings : public settings
  {
  //--- cuerpo de la estructura
  };
```

Al intentar heredar de una estructura con el especificador final, como se muestra en el ejemplo de más arriba, el compilador dará error:

```
cannot inherit from 'settings' as it has been declared as 'final'
see declaration of 'settings'
```

Clases

Las clases llevan una serie de diferencia de las estructuras:

- en la declaración se utiliza la palabra clave class;
- si no se indica lo contrario todos los elementos de las clase por defecto tienen el especificador de acceso private. Los elementos-datos de la estructura por defecto tienen el tipo de acceso public, si no se indica lo contrario;
- los objetos de las clases siempre tienen una tabla de funciones virtuales, incluso si en la clase ninguna función virtual esté declarada. Las estructuras no pueden tener funciones virtuales;
- para los objetos de la clase se puede aplicar el operador new, para las estructuras no se puede aplicar este operador;
- las clases pueden ser heredadas unicamente de las clases, y las estructuras sólo de las estructuras.

© 2000-2025, MetaQuotes Ltd.

Las clases y las estructuras pueden tener el constructor y destructor explícitos. En el caso, si el constructor está determinado de una manera explícita, la inicialización de variable del tipo de la estructura o clase con la ayuda de la sucesión inicializadora es imposible.

Ejemplo:

```
struct trade_settings
  {
   double take;         // valor del precio de fijación del beneficio
   double stop;         // valor del precio del stop de protección
   uchar  slippage;     // valor del deslizamiento permitido
   //--- constructor
          trade_settings() { take=0.0; stop=0.0; slippage=5; }
   //--- destructor
         ~trade_settings() { Print("Es el final"); }
  };
//--- compilador mostrará el error con el mensaje sobre la imposibilidad de inicializa
trade_settings my_set={0.0,0.0,5};
```

Constructores y destructores

El constructor es una función especial que se llama automáticamente cuando se crea un objeto de estructura o clase, y normalmente se utiliza para la inicialización de los miembros de la clase. A continuación vamos a hablar sólo de las clases, pero todo lo dicho también se refiere a las estructuras, si no se especifica lo otro. El nombre del constructor debe coincidir con el de la clase. El constructor no tiene el tipo devuelto (se puede indicar el tipo void).

Algunos miembros definidos de la clase, tales como - cadenas, arrays dinámicos y objetos que requieren la inicialización - de cualquier manera serán inicializados, independientemente de la presencia del constructor.

Cada clase puede tener varios constructores que se diferencian por el número de parámetros y listas de inicialización. Un constructor que se requiere la especificación de parámetros se llama el constructor paramétrico.

Un constructor que no tiene parámetros se llama un **constructor por defecto**. Si en la clase no está declarado ningún constructor, entonces durante la compilación el compilador creará un constructor por defecto.

```
//+------------------------------------------------------------------+
//|  clase para trabajar con la fecha                                |
//+------------------------------------------------------------------+
class MyDateClass
  {
private:
   int               m_year;          // año
   int               m_month;         // mes
   int               m_day;           // día del mes
   int               m_hour;          // hora del día
   int               m_minute;        // minutos
   int               m_second;        // segundos
```

© 2000-2025, MetaQuotes Ltd.

```
public:
   //--- constructor por defecto
                     MyDateClass(void);
   //--- constructor paramétrico
                     MyDateClass(int h,int m,int s);
  };
```

Se puede declarar el constructor en la descripción de la clase y luego definir su cuerpo. Por ejemplo, así se puede definir dos constructores de la clase MyDateClass:

```
//+------------------------------------------------------------------+
//| constructor por defecto                                          |
//+------------------------------------------------------------------+
MyDateClass::MyDateClass(void)
  {
//---
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=mdt.hour;
   m_minute=mdt.min;
   m_second=mdt.sec;
   Print(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| constructor paramétrico                                          |
//+------------------------------------------------------------------+
MyDateClass::MyDateClass(int h,int m,int s)
  {
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=h;
   m_minute=m;
   m_second=s;
   Print(__FUNCTION__);
  }
```

En el constructor por defecto se llenan todos los miembros de la clase por medio de la función TimeCurrent(), en el constructor paramétrico se llenan sólo los valores de la hora. Los demás miembros de la clase (m_year, m_month y m_day) serán inicializados automáticamente con la fecha en curso.

© 2000-2025, MetaQuotes Ltd.

El constructor por defecto tiene un propósito especial cuando se inicializa un array de objetos de su clase. El constructor cuyos parámetros tienen los valores por defecto, **no es** constructor por defecto. Ejemplificaremos esto:

```
//+------------------------------------------------------------------+
//|  clase con un constructor por defecto                            |
//+------------------------------------------------------------------+
class CFoo
  {
   datetime          m_call_time;     // hora de la última llamada al objeto
public:
   //--- un constructor con el parámetro que tiene el valor predefinido no es un const
                     CFoo(datetime t=0){m_call_time=t;};
   string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   CFoo foo; // Esta opción se puede utilizar - se llamará a un constructor con paráme
//--- posibles opciones de creación del objeto CFoo
   CFoo foo1(TimeCurrent());     // la primera opción de creación automática del objet
   CFoo foo2();                  // la segunda opción de creación automática del objet
   CFoo foo3=TimeTradeServer();  // la tercera opción de creación automática del objet
//--- posibles opciones de creación de punteros CFoo por medio del operador new
   CFoo *foo4=new CFoo();
   CFoo *foo5=new CFoo(TimeTradeServer());
   CFoo *foo6=GetPointer(foo5);  // ahora foo5 y foo6 apuntan al mismo objeto
   CFoo *foo7,*foo8;
   foo7=new CFoo(TimeCurrent());
   foo8=GetPointer(foo7);        // foo7 y foo8 apuntan al mismo objeto
   //CFoo foo_array[3];     // esta opción no se puede utilizar - el constructor por d
   //CFoo foo_dyn_array[];  // esta opción no se puede utilizar - el constructor por d
//--- mostramos los valores m_call_time
   Print("foo.m_call_time=",foo1.ToString());
   Print("foo1.m_call_time=",foo1.ToString());
   Print("foo2.m_call_time=",foo2.ToString());
   Print("foo3.m_call_time=",foo3.ToString());
   Print("foo4.m_call_time=",foo4.ToString());
   Print("foo5.m_call_time=",foo5.ToString());
   Print("foo6.m_call_time=",foo6.ToString());
   Print("foo7.m_call_time=",foo7.ToString());
   Print("foo8.m_call_time=",foo8.ToString());
//--- eliminaremos los objetos creados dinámicamente
   delete foo4;
   delete foo5;
   delete foo7;
  }
```

© 2000-2025, MetaQuotes Ltd.

Si añadimos comentarios a estas cadenas en este ejemplo

```
   //CFoo foo_array[3];     // esta opción no se puede utilizar - el constructor por de
```

o

```
   //CFoo foo_dyn_array[];  // esta opción no se puede utilizar - el constructor por de
```

el compilador devolverá el error para ellas "default constructor is not defined".

Si la clase tiene un constructor declarado por el usuario, entonces el compilador no generará el constructor por defecto. Esto quiere decir que si en una clase está declarado un constructor paramétrico pero no está declarado un constructor por defecto, entonces no se puede declarar los arrays de los objetos de esta clase. Pues, para este script el compilador devolverá el error:

```
//+------------------------------------------------------------------+
//| clase sin constructor por defecto                                |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
                     CFoo(string name) { m_name=name;}
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- durante la compilación obtenemos el error "default constructor is not defined"
   CFoo badFoo[5];
  }
```

En este ejemplo la clase CFoo tiene declarado un constructor paramétrico - en este caso durante la compilación el compilador no crea automáticamente el constructor por defecto. Al mismo tiempo, cuando se declara un array de objetos se supone que todos los objetos tienen que ser creados e inicializados automáticamente. Durante la inicialización automática del objeto, es necesario llamar a un constructor por defecto, pero debido a que el constructor por defecto no está declarado explícitamente y no ha sido generado automáticamente por el compilador, entonces resulta imposible crear este objeto. Precisamente por esta razón el compilador muestra el error aún en la fase de compilación.

Existe una sintaxis especial para la inicialización del objeto mediante el constructor. Los inicializadores del constructor (construcciones especiales para la inicialización) para los miembros de una estructura o clase se puede especificar en la lista de inicialización.

La lista de inicialización es una lista de inicializadores separados por comas que sigue tras dos puntos después de la lista de parámetros del constructor y precede el cuerpo (va antes de la llave que abre). Existen algunas exigencias:

- las listas de inicialización se puede utilizar sólo en los constructores;
- no se puede inicializar los miembros de los padres en la lista de inicialización;

© 2000-2025, MetaQuotes Ltd.

- tras las lista de inicialización debe ir la definición (implementación) de la función.

Vamos a mostrar algunos ejemplos de constructores para la inicialización de los miembros de la clase.

```
//+------------------------------------------------------------------+
//| clase para almacenar los apellidos y el nombre de una persona
//+------------------------------------------------------------------+
class CPerson
  {
   string            m_first_name;     // nombre
   string            m_second_name;    // apellido
public:
   //--- constructor por defecto vacío
                     CPerson() {Print(__FUNCTION__);};
   //--- constructor paramétrico
                     CPerson(string full_name);
   //--- constructor con la lista de inicialización
                     CPerson(string surname,string name): m_second_name(surname), m_fi
   void PrintName(){PrintFormat("Name=%s Surname=%s",m_first_name,m_second_name);};
  };
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
CPerson::CPerson(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>=0)
     {
      m_first_name=StringSubstr(full_name,0,pos);
      m_second_name=StringSubstr(full_name,pos+1);
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el error "default constructor is not defined"
   CPerson people[5];
   CPerson Tom="Tom Sawyer";                       // Tom Sawyer
   CPerson Huck("Huckleberry","Finn");             // Huckleberry Finn
   CPerson *Pooh = new CPerson("Whinnie","Pooh");  // Winnie the Pooh
   //--- mostraremos los valores
   Tom.PrintName();
   Huck.PrintName();
   Pooh.PrintName();

   //--- eliminaremos el objeto creado dinámicamente
   delete Pooh;
  }
```

© 2000-2025, MetaQuotes Ltd.

En este caso, la clase CPerson tiene tres constructores:

1. un constructor por defecto explícito que permite crear un array de los objetos de esta clase;
2. un constructor con un parámetro que obtiene el nombre completo como parámetro, y lo divide en el nombre y el apellido según el espacio encontrado;
3. un constructor con dos parámetros que contiene la lista de inicialización. Los inicializadores - m_second_name(surname) y m_first_name(name).

Fíjese cómo la inicialización reemplazó la asignación utilizando la lista. Los miembros individuales deben inicializarse como sigue:

```
miembro_de_la_clase (lista de expresiones)
```

Los miembros pueden seguir cualquier orden en la lista de inicialización, pero todos los miembros de la clase van a inicializarse según el orden de su declaración. Esto significa que en el tercer constructor primero será inicializado el miembro m_first_name porque va declarado primero, y sólo después de él será inicializado el miembro m_second_name. Esto hay que tener en cuenta cuando la inicialización de unos miembros de la clase depende de los valores en otros miembros de la clase.

Si en la clase base no está declarado el constructor por defecto pero al mismo tiempo está declarado uno o varios constructores paramétricos, habrá que llamar sí o sí a uno de los constructores de la clase base en la lista de inicialización. Éste va tras la coma, como los demás miembros de la lista, y será llamado en primer lugar durante la inicialización del objeto independientemente de su ubicación en la lista de inicialización.

```
//+------------------------------------------------------------------+
//|  clase base                                                      |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
   //--- constructor con la lista de inicialización
                     CFoo(string name) : m_name(name) { Print(m_name);}
  };
//+------------------------------------------------------------------+
//|  descendiente de la clase CFoo                                   |
//+------------------------------------------------------------------+
class CBar : CFoo
  {
   CFoo              m_member;      // el miembro de la clase es el objeto del padre
public:
   //--- el constructor por defecto en la lista de inicialización llama al constructor
                     CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   CBar bar;
```

© 2000-2025, MetaQuotes Ltd.

```
   }
```

En el ejemplo mencionado, durante la creación del objeto bar se llamará al constructor por defecto CBar() en el que primero se llama al constructor para el padre CFoo, y luego se llama al constructor para el miembro de la clase m_member.

Los destructores son unas funciones especiales llamadas automáticamente a la hora de eliminar un objeto de la clase. El nombre del destructor se escribe como el de la clase con la tilde (~). Las cadenas, arrays dinámicos y los objetos que necesitan deinicialización van a ser deinicializados de cualquier manera independientemente de la presencia del destructor. Disponiendo del destructor, estas acciones van a ser ejecutadas después del arranque del mismo.

Los destructores siempre son virtuales, independientemente de que si están declarados con la palabra clave virtual o no.

Determinación de los métodos de clase

Las funciones-métodos de clase pueden ser determinados tanto dentro de la clase, como fuera de la declaración de la clase. Si el método se determina dentro de la clase, entonces su cuerpo sigue directamente después de la declaración del método.

Ejemplo:

```
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
   void              Rotate()                   { m_prev_turn=m_turn; if(++m_turn>3) r
   virtual void      Draw()                     { return;                }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

Las funciones con SetRightBorder(int border) por Draw() se declaran y se determinan directamente dentro de la clase CTetrisShape.

© 2000-2025, MetaQuotes Ltd.

El constructor CTetrisShape() y los métodos CheckDown(int& pad_array[]), CheckLeft(int& side_row[]) y CheckRight(int& side_row[]) se declaran sólo dentro de la clase, pero por ahora no están determinados. Las determinaciones de estas funciones deben seguir más adelante en el código. Para determinar el método fuera de la clase se utiliza la operación del permiso de contexto, como contexto se utiliza el nombre de la clase.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Constructor de la clase base                                     |
//+------------------------------------------------------------------+
void CTetrisShape::CTetrisShape()
  {
   m_type=0;
   m_ypos=0;
   m_xpos=0;
   m_xsize=SHAPE_SIZE;
   m_ysize=SHAPE_SIZE;
   m_prev_turn=0;
   m_turn=0;
   m_right_border=0;
  }
//+------------------------------------------------------------------+
//| Prueba de posibilidad de moverse abajo (para la vara o el cubo)  |
//+------------------------------------------------------------------+
bool CTetrisShape::CheckDown(int& pad_array[])
  {
   int i,xsize=m_xsize/SHAPE_SIZE;
//---
   for(i=0; i<xsize; i++)
     {
      if(m_ypos+m_ysize>=pad_array[i]) return(false);
     }
//---
   return(true);
  }
```

Especificadores de acceso public, protected y private

A la hora de crear nueva clase se recomienda limitar el acceso a los elementos desde fuera. Para eso se utilizan las palabras claves private o protected. En este caso el acceso a los datos encubiertos puede realizarse sólo desde las funciones-métodos de la misma clase. Si se utiliza la palabra clave protected, entonces el acceso a los datos encubiertos se puede realizar también de los métodos de las clases que son herederos de esta clase. De la misma manera se puede limitar el acceso a las funciones-métodos de clase.

Si se necesita abrir totalmente el acceso a los elementos y/o métodos de clase, entonces se utiliza la palabra clave public.

Ejemplo:

```
class CTetrisField
  {
private:
   int               m_score;                        // cuenta
```

© 2000-2025, MetaQuotes Ltd.

```
   int               m_ypos;                              // posición actual de la piez
   int               m_field[FIELD_HEIGHT][FIELD_WIDTH]; // matrix del vaso
   int               m_rows[FIELD_HEIGHT];                // numeración de las filas de
   int               m_last_row;                          // la última fila libre
   CTetrisShape     *m_shape;                             // pieza del tetris
   bool              m_bover;                             // fin del juego
public:
   void              CTetrisField() { m_shape=NULL; m_bover=false; }
   void              Init();
   void              Deinit();
   void              Down();
   void              Left();
   void              Right();
   void              Rotate();
   void              Drop();
private:
   void              NewShape();
   void              CheckAndDeleteRows();
   void              LabelOver();
  };
```

Cualquier elemento y método de la clase que están declarados después del especificador public (y hasta el siguiente especificador del acceso), son accesibles a la hora de cualquier referencia del programa hacia el objeto de esta clase. En este ejemplo son los siguientes elementos: funciones CTetrisField(), Init(), Deinit(), Down(), Left(), Right(), Rotate() y Drop().

Cualquier elemento de la clase que está declarado después del especificador private (y hasta el siguiente especificador del acceso), es accesible sólo para las funciones-elementos de la misma clase. Los especificadores de acceso a los elementos siempre se terminan con dos puntos (:) y pueden aparecer en la determinación de la clase varias veces.

Cualquier miembro de clase declarado después del especificador de acceso **protected:** (y hasta el próximo identificador de acceso) solo estará disponible para las funciones de miembro de esta clase y las funciones de miembro de los herederos de esta clase. Si intentamos recurrir a los miembros con especificadores private y protected desde fuera, obtendremos un error del estadio de compilación. Ejemplo:

```
class A
  {
protected:
   //--- el operador de copiado está disponible solo desde dentro de la clase A y sus
   void operator=(const A &)
     {
     }
  };
class B
  {
   //--- declarado un objeto de clase A
   A                 a;
  };
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- declaramos dos variables del tipo B
   B b1, b2;
   //--- intentando copiar un objeto en otro
   b2=b1;
  }
```

Al compilar este código, obtendremos un mensaje de error sobre el intento de llamar el operador de copiado remoto:

```
attempting to reference deleted function 'void B::operator=(const B&)'   trash3.mq5
```

En la segunda línea se nos ofrecerá una descripción más detallada: el operador de copiado en la clase B ha sido eliminado explícitamente, dado que se llama al operador de copiado de la clase A, que no está disponible:

```
   function 'void B::operator=(const B&)' was implicitly deleted because it invokes ir
```

El acceso a los elementos de la clase base puede volver a determinarse durante la herencia en las clases derivadas.

El especificador delete

El especificador delete marca las funciones de miembro de la clase que no se pueden utilizar. Esto significa que si un programa recurre de forma explícita o implícita a una función así, obtendremos un error ya en la etapa de compilación. Por ejemplo, este especificador permite hacer inaccesible los métodos padre en la clase hija. El mismo resultado podemos conseguir si declaramos la función en la en la zona privada de la clase padre (declaración en la sección private). El uso de delete en este caso hace el código más legible y gestionable al nivel de los herederos.

```
class A
  {
public:
                     A(void) {value=5;};
   double            GetValue(void) {return(value);}
private:
   double            value;
  };
class B: public A
  {
   double            GetValue(void)=delete;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos la variable de tipo A
```

© 2000-2025, MetaQuotes Ltd.

```
   A a;
   Print("a.GetValue()=", a.GetValue());
//--- intentando obtener el valor de la variable de tipo B
   B b;
   Print("b.GetValue()=", b.GetValue()); // el compilador dará error en esta línea
  }
```

Mensaje del compilador:

```
attempting to reference deleted function 'double B::GetValue()'
   function 'double B::GetValue()' was explicitly deleted here
```

Con la ayuda del especificador delete se puede prohibir la conversión automática de tipos o el constructor de copias, que de lo contrario deberíamos ocultar también en la sección private. Ejemplo:

```
class A
  {
public:
   void              SetValue(double v) {value=v;}
   //--- prohibimos la llamada con el tipo int
   void              SetValue(int) = delete;
   //--- prohibimos el operador de copiado
   void              operator=(const A&) = delete;
private:
   double            value;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos dos variables del tipo A
   A a1, a2;
   a1.SetValue(3);       // ¡error!
   a1.SetValue(3.14);    // OK
   a2=a1;                // ¡error!
  }
```

Al intentar compilar, obtenemos un mensaje sobre los siguientes errores:

```
attempting to reference deleted function 'void A::SetValue(int)'
   function 'void A::SetValue(int)' was explicitly deleted here
attempting to reference deleted function 'void A::operator=(const A&)'
   function 'void A::operator=(const A&)' was explicitly deleted here
```

El especificador final

La presencia del especificador final al declarar la clase, prohíbe la posterior herencia del mismo. Si la interfaz de la clase está construida de tal forma que no hay necesidad de introducir en el mismo cambios posteriores, o los cambios no están permitidos por motivos de seguridad, deberemos declarar

© 2000-2025, MetaQuotes Ltd.

la clase con el especificador "final". En este caso, además, todos los métodos de la clase se considerarán "final" de forma implícita.

```
class CFoo final
  {
  //--- cuerpo de la clase
  };

class CBar : public CFoo
  {
  //--- cuerpo de la clase
  };
```

Al intentar heredar de una clase con el especificador "final", el compilador dará error, como se muestra en el ejemplo de más arriba:

```
cannot inherit from 'CFoo' as it has been declared as 'final'
see declaration of 'CFoo'
```

Uniónes (union)

Una unión supone un tipo especial de datos que consta de varias variables que comparten una misma zona de la memoria. Por consiguiente, la unión proporciona la posibilidad de interpretar una misma secuencia de bits con dos (o más) métodos diferentes. La declaración de una unión es semejante a la declaración de una estructura y comienza con la palabra clave union.

```
union LongDouble
{
  long   long_value;
  double double_value;
};
```

Pero, a diferencia de la estructura, los diferentes miembros de una unión se relacionan con una misma zona de la memoria. En este ejemplo se ha declarado la unión LongDouble, en la que el valor del tipo long y el valor del tipo double comparten la misma zona de la memoria. Es importante comprender que no es posible hacer que la unión guarde al mismo tiempo valores de tipo entero *long* y reales *double* (como sucedía en la estructura), puesto que las variables long_value y double_value se solapan (en la memoria) una sobre otra. Sin embargo, un programa MQL5 puede en cualquier momento procesar la información que se contiene en esta unión como un valor entero (long) o como uno real (double). Por consiguiente, la unión permite obtener dos (o más) variantes de representación de una misma secuencia de datos.

Al declarar una unión, el compilador delimita automáticamente una parte de la memoria que sea suficiente para guardar en una unión las variables del tipo de volumen más grande. Para acceder a un elemento de la unión, se usa la misma sintaxis que para las estructuras: el operador "punto".

```
union LongDouble
{
  long   long_value;
  double double_value;
};
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//---
   LongDouble lb;
//--- obtenemos un número no válido -nan(ind) y lo mostramos
   lb.double_value=MathArcsin(2.0);
   printf("1.  double=%f                integer=%I64X",lb.double_value,lb.long_value);
//--- el mayor número normalizado (DBL_MAX)
   lb.long_value=0x7FEFFFFFFFFFFFFF;
   printf("2.  double=%.16e  integer=%I64X",lb.double_value,lb.long_value);
//--- el menor número normalizado positivo (DBL_MIN)
   lb.long_value=0x0010000000000000;
   printf("3.  double=%.16e  integer=%.16I64X",lb.double_value,lb.long_value);
  }
/*  Resultado de la ejecución
    1.  double=-nan(ind)                integer=FFF8000000000000
    2.  double=1.7976931348623157e+308  integer=7FEFFFFFFFFFFFFF
    3.  double=2.2250738585072014e-308  integer=0010000000000000
*/
```

Puesto que las uniones permiten al programa interpretar los mismos datos en la memoria de forma diferente, con frecuencia se usan en los casos en que se necesita una conversión de tipos.

Las uniones no pueden participar en la herencia, y tampoco pueden tener miembros estáticos por definición. Por lo demás, *union* se comporta como una estructura en la que todos los miembros tienen un desplazamiento cero. Al mismo tiempo, no pueden ser miembros de una unión los siguientes tipos:

- matrices dinámicas
- líneas de caracteres
- punteros a objetos y funciones
- objetos de clases
- objetos de estructuras que tengan constructores o destructores
- objetos de estructuras que contengan miembros de los puntos 1-5

Al igual que las clases, una unión puede tener constructores y destructores, y también los métodos. Por defecto, los miembros de una unión tienen un tipo de acceso public, para crear elementos cerrados, es necesario usar la palabra clave private. Todas estas posibilidades se muestran en este ejemplo, que representa cómo convertir un color con el tipo color en una representación ARGB, como hace la función ColorToARGB().

```
//+------------------------------------------------------------------+
//| Unión para la conversión de color(BGR) en una representación ARGB |
//+------------------------------------------------------------------+
union ARGB
  {
   uchar             argb[4];
   color             clr;
   //--- constructores
                     ARGB(color col,uchar a=0){Color(col,a);};
                    ~ARGB(){};
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- métodos públicos
public:
   uchar    Alpha(){return(argb[3]);};
   void     Alpha(const uchar alpha){argb[3]=alpha;};
   color    Color(){ return(color(clr));};
   //--- métodos privados
private:
   //+------------------------------------------------------------------+
   //| configuración del color y el valor del canal alfa                |
   //+------------------------------------------------------------------+
   void     Color(color col,uchar alpha)
     {
      //--- establecemos el color en el miembro clr
      clr=col;
      //--- establecemos el valor del componente Alpha - nivel de opacidad
      argb[3]=alpha;
      //--- cambiamos de lugar los bytes de los componentes R y B (Red y Blue)
      uchar t=argb[0];argb[0]=argb[2];argb[2]=t;
     };
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- el valor 0x55 designa 55/255=21.6 % (0% - totalmente transparente)
   uchar alpha=0x55;
//--- el tipo "color" tiene una representación 0x00BBGGRR
   color test_color=clrDarkOrange;
//--- aquí aplicaremos los valores de los bytes de la unión ARGB
   uchar argb[];
   PrintFormat("0x%.8X - este es el aspecto del tipo "color" para %s, BGR=(%s)",
               test_color,ColorToString(test_color,true),ColorToString(test_color));
//--- el tipo ARGB se representa como 0x00RRGGBB, los componentes RR y BB han sido car
   ARGB argb_color(test_color);
//--- copiamos la matriz de bytes
   ArrayCopy(argb,argb_color.argb);
//--- veamos qué aspecto tiene en la representación ARGB
   PrintFormat("0x%.8X - representación ARGB con un canal alfa=0x%.2x, ARGB=(%d,%d,%d,
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- añadimos el valor de opacidad
   argb_color.Alpha(alpha);
//--- intentamos mostrar ARGB como el tipo "color"
   Print("ARGB como color=(",argb_color.clr,")  canal alfa=",argb_color.Alpha());
//--- copiamos la matriz de bytes
   ArrayCopy(argb,argb_color.argb);
//--- este es el aspecto que tiene en la representación ARGB
   PrintFormat("0x%.8X - representación ARGB con un canal alfa=0x%.2x, ARGB=(%d,%d,%d,
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- lo cotejamos con lo mostrado por la función ColorToARGB()
   PrintFormat("0x%.8X - resultado ColorToARGB(%s,0x%.2x)",ColorToARGB(test_color,alph
               ColorToString(test_color,true),alpha);
  }
/* Resultado de la ejecución
   0x00008CFF - este es el aspecto del tipo color para clrDarkOrange, BGR=(255,140,0)
   0x00FF8C00 - representación ARGB con un canal alfa=0x00, ARGB=(0,255,140,0)
   ARGB como color=(0,140,255)  canal alfa=85
   0x55FF8C00 - representación ARGB con un canal alfa=0x55, ARGB=(85,255,140,0)
   0x55FF8C00 - resultado ColorToARGB(clrDarkOrange,0x55)
*/
```

Interfaces

La interfaz ha sido diseñada para definir una cierta funcionalidad, cuya clase en consecuencia puede implementar. En la práctica, se trata de una clase que no puede contener miembros y puede tener un constructor y/o destructor. Todos los métodos declarados en la interfaz son puramente virtuales, incluso sin definición explícita.

Se define la interfaz con la ayuda de la palabra clave interface, como se muestra en el ejemplo:

```
//--- interfaz básica para describir animales
interface IAnimal
  {
//--- los métodos de la interfaz, por defecto, tienen acceso public
   void Sound();  // sonido que hace el animal
  };
//+------------------------------------------------------------------+
//|  la clase CCat se hereda de la interfaz IAnimal                  |
//+------------------------------------------------------------------+
class CCat : public IAnimal
  {
public:
                     CCat() { Print("Cat was born"); }
                    ~CCat() { Print("Cat is dead");  }
   //--- implementamos el método Sound de la interfaz IAnimal
   void Sound(){ Print("meou"); }
  };
//+------------------------------------------------------------------+
//|  la clase CDog se hereda de la interfaz IAnimal                  |
//+------------------------------------------------------------------+
class CDog : public IAnimal
  {
public:
                     CDog() { Print("Dog was born"); }
                    ~CDog() { Print("Dog is dead");  }
   //--- implementamos el método Sound de la interfaz IAnimal
   void Sound(){ Print("guaf"); }
  };
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- matriz de punteros a los objetos del tipo IAnimal
   IAnimal *animals[2];
//--- generamos los descendientes de IAnimal y guardamos los punteros a estos en una r
   animals[0]=new CCat;
   animals[1]=new CDog;
//--- llamamos el método Sound() de la interfaz básica IAnimal para cada descendiente
   for(int i=0;i<ArraySize(animals);++i)
      animals[i].Sound();
//--- eliminamos los objetos
   for(int i=0;i<ArraySize(animals);++i)
      delete animals[i];
//--- resultado de la ejecución
/*
   Cat was born
   Dog was born
   meou
   guaf
   Cat is dead
   Dog is dead
*/
  }
```

Como sucede con las clases abstractas, no se puede crear un objeto de la interfaz sin herencia. La interfaz puede heredarse solo de otras interfaces y puede actuar como descendiente para la clase. Además, siempre tiene visibilidad pública.

La interfaz no se puede declarar dentro de la declaración de una clase o estructura, pero así y con todo, el puntero a la interfaz se puede guardar en una variable del tipo void *. Hablando en general, en una variable del tipo void * se puede guardar un puntero a un objeto de cualquier clase. Para transformar el puntero void * en el puntero a un objeto de una clase concreta, es necesario usar el operador dynamic_cast. En el caso de que la transformación no sea posible, el resultado de la operación dynamic_cast será NULL.

Véase también

Programación orientada a objetos

© 2000-2025, MetaQuotes Ltd.

Objeto de un array dinámico

Arrays dinámicos

Se permite declarar un array que no no supere 4 dimensiones. Al declarar una matriz dinámica (matriz sin valor expreso en el primer par de los corchetes), el compilador crea automáticamente una variable de la estructura indicada más arriba (objeto de matriz dinámica) y proporciona el código para una inicialización correcta.

Los arrays dinámicos se liberan automáticamente, al salir fuera de los límites de la visibilidad del bloque donde han sido declarados.

Ejemplo:

```
double matrix[][10][20]; // array dinámica de 3 dimensiones
ArrayResize(matrix,5);   // indicamos el tamaño de la primera dimensión
```

Arrays estáticos

Si todas las dimensiones significativas de un array se indican de una manera explícita, el compilador de antemano distribuye el espacio necesario de la memoria. Este array se llama array estático. No obstante, el compilador distribuye adicionalmente la memoria para el objeto del array dinámico. Este objeto está vinculado con el búfer estático distribuido previamente (sector de memoria para almacenamiento de la matriz).

La creación de un objeto del array dinámico está condicionado a la posible necesidad de transmitir dicho array estático como un parámetro a alguna función.

Ejemplos:

```
double stat_array[5]; // array dinámico de 1 dimensión
some_function(stat_array);
...
bool some_function(double& array[])
  {
   if(ArrayResize(array,100)<0) return(false);
   ...
   return(true);
  }
```

Arrays como partes de las estructuras

Al declarar un array estático como un elemento de una estructura, el objeto del array dinámico no se crea. Esto se hace para la compatibilidad de las estructuras de datos que se utilizan en Windows API.

Sin embargo, los arrays estáticos declarados como elementos de las estructuras, también se puede pasar a las funciones MQL5. En este caso, al pasar el parámetro va a crearse un objeto de un array dinámico temporal vinculado con el array estático - elemento de la estructura.

Véase también

© 2000-2025, MetaQuotes Ltd.

Operaciones con arrays, Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Matrices y vectores

El tipo vector es un tipo de datos especial en MQL5 que permite realizar operaciones con vectores. Un vector es un array (matriz) unidimensional de tipo double. Es uno de los conceptos fundamentales del álgebra lineal, utilizado en muchos campos de la ciencia, incluida la física, la geometría y otros. Los vectores se usan para resolver sistemas de ecuaciones lineales, en gráficos 3D y en otras áreas de aplicación. Los vectores se pueden sumar y multiplicar. La longitud o distancia entre vectores se puede obtener mediante la Norma. En programación, los vectores se representan generalmente con la ayuda de arrays de elementos homogéneos que pueden no tener operaciones vectoriales regulares, es decir, cuando los arrays no se pueden sumar o multiplicar, y no tienen una norma.

Los vectores se pueden representar como vectores de fila y vectores de cadena cuando se trabaja con matrices. Además, los vectores en álgebra lineal utilizan los conceptos de covarianza y contravarianza. Estos conceptos no suponen ninguna diferencia a la hora de escribir un código en MQL5, ya que solo el programador decide qué es cada objeto del tipo vectorial. Por ejemplo, puede ser vector de rotación, de desplazamiento o de compresión en gráficos 3D.

En términos generales, desde el punto de vista del álgebra lineal, un número también es un vector, pero en un espacio vectorial unidimensional. Un vector en sí mismo puede considerarse como un caso especial de una matriz.

El tipo matrix es otro tipo de datos especial en MQL5 para representar matrices. Una matriz es en realidad un array bidimensional del tipo double. Los vectores y matrices se han introducido en MQL5 para facilitar las operaciones con ciertos tipos de conjuntos de datos. Con ellos, los desarrolladores pueden beneficiarse de las posibilidades del álgebra lineal de una forma sencilla y similar a las matemáticas. Las matrices se pueden usar para escribir sistemas de ecuaciones diferenciales o lineales de forma compacta. El número de filas de la matriz se corresponde con el número de ecuaciones, mientras que el número de columnas es igual al número de incógnitas. Como resultado, los sistemas de ecuaciones lineales se pueden resolver utilizando operaciones con matrices.

Existen los siguientes tipos de datos:

- matrix — matriz que contiene elementos de tipo double
- matrixf — matriz que contiene elementos del tipo float
- matrix — matriz que contiene elementos del tipo complex
- vector — vector que contiene elementos del tipo double
- vectorf — vector que contiene elementos del tipo float
- vectorc — vector que contiene elementos del tipo complex

Para su uso en funciones de plantilla, podemos utilizar la entrada matrix<double>, matrix<float>, matrix<complex>, vector<double>, vector<float>, vector<complex> en lugar de los tipos correspondientes.

Para las matrices se definen las siguientes operaciones algebraicas:

- Adición de matrices del mismo tamaño
- Multiplicación de matrices de tamaño adecuado: el número de columnas en la matriz izquierda debe ser igual al número de filas en la matriz derecha
- Multiplicación de matrices por un vector de columna; multiplicación de un vector de fila por una matriz según la regla de multiplicación de matrices. En este sentido, el vector supone un caso especial de una matriz.

© 2000-2025, MetaQuotes Ltd.

- Multiplicación de matrices por un número, es decir, por un escalar

Las matemáticas consideran muchos tipos de matrices distintos. Por ejemplo, matrices de identidad, simétricas, asimétricas, triangulares superior e inferior y de otros tipos. Varias formas normales juegan un papel importante en la teoría de matrices. Representan una determinada forma canónica de una matriz, que se puede obtener con la ayuda de determinadas transformaciones. En la práctica, se usan formas normales con propiedades adicionales, como por ejemplo la estabilidad.

El uso de vectores y matrices, o mejor dicho, de métodos especiales de los tipos relevantes, permite crear un código más simple, claro y conciso, cercano a la notación matemática. Con estos métodos, podrá evitar la necesidad de crear ciclos anidados o tener en cuenta la indexación correcta de arrays en los cálculos. Por consiguiente, la utilización de estos métodos aumenta la fiabilidad y la velocidad de desarrollo de programas complejos.

Lista de métodos de matrices y vectoriales

Los tipos matrix y vector incluyen métodos que se corresponden con los métodos relevantes de la biblioteca NumPy. Con estos métodos, podrá traducir algoritmos y códigos de Python a MQL5 con un esfuerzo mínimo. Muchas tareas de procesamiento de datos, ecuaciones matemáticas, redes neuronales y tareas de aprendizaje automático se pueden resolver con la ayuda de bibliotecas y métodos de Python listos para usar.

Método matrix/vector	Método análogo en NumPy	Descripción
void matrix.Eye(const int rows, const int cols, const int ndiag=0)	eye	Construye una matriz con unidades en la diagonal y ceros en el resto
void matrix.Identity(const int rows)	identity	Construye una matriz cuadrada con unidades en la diagonal principal
void matrix.Ones(const int rows, const int cols)	ones	Construye una nueva matriz de filas y columnas dadas, rellenadas con unidades
void matrix.Zeros(const int rows, const int cols)	zeros	Construye una nueva matriz de filas y columnas dadas, rellenadas con ceros
void matrix.Full(const int rows, const int cols, const scalar value)	full	Construye una nueva matriz de filas y columnas dadas, rellenadas con un valor escalar
void matrix.Copy(const matrix a)	copy	Construye una copia de la matriz dada
void matrix.FromBuffer(const int rows, const int cols, const scalar array[], const int count=-1, const int offset=0)	frombuffer	Construye una matriz creada a partir de un array unidimensional
void matrix.FromFile(const int rows, condt int cols, const int file_handle,	fromfile	Construye una matriz a partir de los datos en un archivo de texto o

© 2000-2025, MetaQuotes Ltd.

Método matrix/vector	Método análogo en NumPy	Descripción
const int count=-1, const int offset=0)		binario
void vector.FromString(const string source, const string sep=" ")	fromstring	Construye un vector inicializado a partir de los datos de texto en una cadena
void vector.Arange(const scalar start, const scalar stop, const scalar step=1)	arange	Construye valores espaciados uniformemente dentro de un intervalo dado
void matrix.Diag(const vector v, const int ndiag=0)	diag	Extrae una diagonal o construye una matriz diagonal
void matrix.Tri(const int rows, const int cols, const int ndiag=0)	tri	Construye una matriz con unidades en la diagonal dada y por debajo de esta, y ceros en cualquier otro lugar
void matrix.Tril(const int rows, const int cols, const scalar array[], const int ndiag=0)	tril	Retorna una copia de una matriz con elementos puestos a cero por encima de la k-ésima diagonal
void matrix.Triu(const int rows, const int cols, const scalar array[], const int ndiag=0)	triu	Retorna una copia de una matriz con elementos puestos a cero por debajo de la k-ésima diagonal
void matrix.Vander(const vector v, const int cols=-1, const bool increasing=false)	vander	Genera una matriz de Vandermonde
vector matrix.Row(const unsigned nrow)		Retorna un vector de fila
vector matrix.Col(const unsigned ncol)		Retorna un vector de columna
unsigned matrix.Rows()		Retorna el número de filas en una matriz
unsigned matrix.Cols()		Retorna el número de columnas en una matriz
void matrix.Init()		Inicializa una matriz
matrix matrix.Transpose()	transpose	Invierte o permuta los ejes de una matriz; retorna la matriz modificada
matrix matrix.Dot(const matrix b)	dot	Producto escalar de dos matrices
matrix matrix.Inner(const matrix b)	inner	Producto interno de dos matrices
matrix matrix.Outer(const matrix b)	outer	Calcula el producto externo de dos matrices

© 2000-2025, MetaQuotes Ltd.

Método matrix/vector	Método análogo en NumPy	Descripción
matrix matrix.MatMul(const matrix b)	matmul	Producto matricial de dos matrices
matrix matrix.MatrixPower(const int power)	matrix_power	Eleva una matriz cuadrada a la potencia (entera) n
matrix matrix.Kron(const matrix b)	kron	Retorna el producto de Kronecker de dos matrices
bool matrix.Cholesky(matrix& L)	cholesky	Retorna la descomposición de Cholesky
bool matrix.QR(matrix& Q, matrix& R)	qr	Calcula la factorización qr de una matriz
bool matrix.SVD(matrix& U, matrix& V, vector& singular_values)	svd	Valor singular de descomposición
bool matrix.Eig(matrix& eigen_vectors, vector& eigen_values)	eig	Calcula los valores propios y los vectores propios derechos de una matriz cuadrada
bool matrix.EigH(matrix& eigen_vectors, vector& eigen_values)	eigh	Retorna los valores propios y vectores propios de una matriz hermitiana
bool matrix.EigVals(vector& eigen_values)	eigvals	Calcula los valores propios de una matriz general
bool matrix.EigValsH(vector& eigen_values)	eigvalsh	Calcula los valores propios de una matriz hermitiana
bool matrix.LU(matrix& L, matrix& U)		Descomposición LU de una matriz como el producto de una matriz triangular inferior y una matriz triangular superior
bool matrix.LUP(matrix& L, matrix& U, matrix& P)		Descomposición LUP con pivoteo parcial, referido a la descomposición LU con permutaciones de fila únicamente:PA=LU
double matrix.Norm(const norm)	norm	Retorna la norma de una matriz o vector
double matrix.Cond(const norm)	cond	Calcula el número de condición de una matriz
vector matrix.Spectrum()		Calcula el espectro de una matriz como el conjunto de sus valores propios del producto AT*A
double matrix.Det()	det	Calcula el determinante de un array

© 2000-2025, MetaQuotes Ltd.

Método matrix/vector	Método análogo en NumPy	Descripción
int matrix.Rank()	matrix_rank	Retorna el rango de matriz de un array usando el método gaussiano
int matrix.SLogDet(int& sign)	slogdet	Calcula el signo y el logaritmo del determinante de un array
double matrix.Trace()	trace	Retorna la suma a lo largo de las diagonales de la matriz
vector matrix.Solve(const vector b)	solve	Resuelve una ecuación de matriz lineal o un sistema de ecuaciones algebraicas lineales
vector matrix.LstSq(const vector b)	lstsq	Retorna la solución de los mínimos cuadrados de ecuaciones algebraicas lineales (para matrices no cuadradas o degeneradas)
matrix matrix.Inv()	inv	Calcula la inversa (multiplicativa) de una matriz
matrix matrix.PInv()	pinv	Calcula la pseudo-inversa de una matriz con el método de Moore-Penrose
int matrix.Compare(const matrix matrix_c, const double epsilon) int matrix.Compare(const matrix matrix_c, const int digits) int vector.Compare(const vector vector_c, const double epsilon) int vector.Compare(const vector vector_c, const int digits)		Compara los elementos de dos matrices/vectores con la precisión indicada
double matrix.Flat(const ulong index) bool matrix.Flat(const ulong index,const double value)	flat	Permite dirigir un elemento de la matriz a través de un índice, en lugar de dos
double vector.ArgMax() double matrix.ArgMax() vector matrix.ArgMax(const int axis)	argmax	Retorna el índice del valor máximo
double vector.ArgMin() double matrix.ArgMin() vector matrix.ArgMin(const int axis)	argmin	Retorna el índice del valor mínimo
double vector.Max() double matrix.Max() vector matrix.Max(const int axis)	max	Retorna el valor máximo en una matriz/vector

© 2000-2025, MetaQuotes Ltd.

Método matrix/vector	Método análogo en NumPy	Descripción
double vector.Mean() double matrix.Mean() vector matrix.Mean(const int axis)	mean	Calcula la media aritmética de los valores de los elementos
double vector.Min() double matrix.Min() vector matrix.Min(const int axis)	min	Retorna el valor mínimo en una matriz/vector
double vector.Sum() double matrix.Sum() vector matrix.Sum(const int axis)	sum	Ejecuta la suma de los elementos de una matriz/vector, que también se puede realizar según el eje(s) indicado(s).
void vector.Clip(const double min_value,const double max_value) void matrix.Clip(const double min_value,const double max_value)	clip	Limita los elementos de una matriz/vector con el intervalo indicado de valores permitidos
vector vector.CumProd() vector matrix.CumProd() matrix matrix.CumProd(const int axis)	cumprod	Retorna el producto acumulado de los elementos de un array/vector, incluido en el eje indicado
vector vector.CumSum() vector matrix.CumSum() matrix matrix.CumSum(const int axis)	cumsum	Retorna la suma acumulada de los elementos de una matriz/vector, incluido en el eje indicado
double vector.Prod(const double initial=1) double matrix.Prod(const double initial=1) vector matrix.Prod(const int axis,const double initial=1)	prod	Calcula el producto de los elementos de una matriz/vector, que también se puede ejecutar según el eje indicado
void matrix.Reshape(const ulong rows, const ulong cols)	reshape	Cambia la forma de una matriz sin modificar sus datos
void matrix.Resize(const ulong rows,const ulong cols)	resize	Retorna una nueva matriz con la forma indicada
bool matrix.SwapRows(const ulong row1, const ulong row2)		Muda de sitio las líneas en una matriz
bool matrix.SwapCols(const ulong col1, const ulong col2)		Muda de sitio las columnas en una matriz
double vector.Ptp() double matrix.Ptp() vector matrix.Ptp(const int axis)	ptp	Retorna el intervalo de valores de una matriz/vector o del eje indicado de una matriz y del equivalente al resultado Max() - Min()

© 2000-2025, MetaQuotes Ltd.

Método matrix/vector	Método análogo en NumPy	Descripción
double vector.Percentile(const int percent) double matrix.Percentile(const int percent) vector matrix.Percentile(const int percent,const int axis)	percentile	Calcula el percentil indicado de los valores de los elementos de una matriz/vector o de los elementos a lo largo del eje indicado. Los valores permitidos del parámetro percent se encuentran en el intervalo [0, 100]
double vector.Quantile(const int percent) double matrix.Quantile(const int percent) vector matrix.Quantile(const int percent,const int axis)	quantile	Calcula el cuantil indicado de los valores de los elementos de una matriz/vector o de los elementos a lo largo del eje indicado. El intervalo de valores del parámetro percent adopta los valores en el intervalo [0, 1]
double vector.Median() double matrix.Median() vector matrix.Median(const int axis)	median	Calcula la mediana de los elementos de una matriz/vector. La mediana es un valor tal que exactemente la mitad de los elementos de una matriz/vector resultará menor a él, mientras que la otra será mayor.
double vector.Average() double matrix.Average() vector matrix.Average(const int axis)	average	Calcula la media aritmética ponderada de los elementos de una matriz/vector. La suma de los pesos en el denominador no puede ser igual a 0, pero ciertos pesos pueden ser iguales a 0
double vector.Std() double matrix.Std() vector matrix.Std(const int axis)	std	Calcula la desviación media cuadrática (estándar) de los valores de los elementos de una matriz/vector o del eje indicado
double vector.Var() double matrix.Var() vector matrix.Var(const int axis)	var	Calcula la varianza de los valores de los elementos de una matriz/vector
double vector.CorrCoef(const vector& v) matrix matrix.CorrCoef()	corrcoef	Calcula el coeficiente de correlación de Pearson (coeficiente de correlación lineal). El coeficiente de correlación se encuentra en el intervalo [-1, 1]
vector vector.Correlate(const vector& v,enum mode)	correlate	Calcula los valores de la función de correlación mutua (correlación cruzada) de dos vectores. El parámetro "mode" determina el modo de cálculo de la convolución lineal

© 2000-2025, MetaQuotes Ltd.

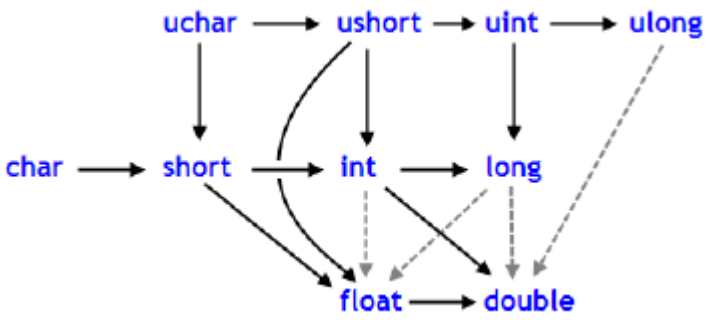
Método matrix/vector	Método análogo en NumPy	Descripción
vector vector.Convolve(const vector& v, enum mode)	convolve	Retorna la convolución lineal discreta de dos vectores El parámetro "mode" determina el modo de cálculo de la convolución lineal
matrix matrix.Cov() matrix vector.Cov(const vector& v); (matriz resultante 2 x 2)	cov	Calcula la matriz de covarianza. La covarianza de dos muestras (dos magnitudes aleatorias) es la medida de su dependencia lineal

© 2000-2025, MetaQuotes Ltd.

Conversión de tipos

Transformación de tipos numéricos

Muy a menudo surge la necesidad de transformar un tipo de datos en otro. No cada uno de los tipos numéricos puede ser transformado en otro. Las transformaciones admisibles en MQL5 se indican en el esquema:



Las líneas contínuas con flechas indican las transformaciones que se efectuan sin pérdida de la información. El tipo bool puede figurar en vez del tipo char (ambos ocupan 1 byte de la memoria), en vez del tipo int se puede utilizar el tipo color (4 bytes cada uno), y en vez del tipo long se admite el tipo datetime (ocupan 8 bytes cada uno). Cuatro líneas rayadas con flechas de color gris significan las transformaciones en las que puede ocurrirse la pérdida de precisiones. Por ejemplo, la cantidad de dígitos en el número entero 123456789 (int) supera la cantidad de dígitos que puede ser representada en el tipo float.

```
   int n=123456789;
   float f=n;     // contenido f es igual a 1.234567892E8
   Print("n = ",n,"   f = ",f);
   // resultado n= 123456789    f= 123456792.00000
```

El número transformado en el tipo float tiene el mismo orden pero con una precisión algo menor. Las transformaciones inversas a las flechas negras se realizan con una posible pérdida de la información. Las transformaciones entre char y uchar, short y ushort, int y uint, long y ulong (se tienen en cuenta las transformaciones directas e inversas) pueden llevar a la pérdida de la información.

Resulta que al transformar los valores con punto flotante en el tipo entero, la parte fraccionaria siempre se omite. Si se necesita redondear el número con punto flotante hasta un número entero más próximo (lo que en mayoría de los casos puede ser más útil) es necesario utilizar la función MathRound().

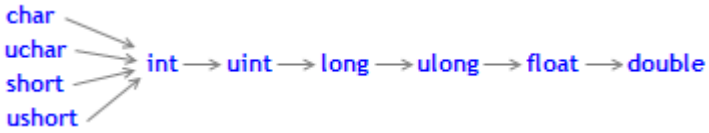
Ejemplo:

```
//--- aceleración de la gravedad
   double g=9.8;
   double round_g=(int)g;
   double math_round_g=MathRound(g);
   Print("round_g = ",round_g);
   Print("math_round_g = ",math_round_g);
/*
   Resultado:
```

© 2000-2025, MetaQuotes Ltd.

```
   round_g = 9
   math_round_g = 10
*/
```

Si dos valores se unen con un operador binario, entonces antes de iniciar la operación el operando del tipo menor se transforma en el tipo mayor según la prioridad indicada en la esquema:



En las operaciones los tipos de datos char, uchar, short y ushort se convierten al tipo int incondicionalmente.

Ejemplos:

```
   char   c1=3;
//--- primer ejemplo
   double d2=c1/2+0.3;
   Print("c1/2 + 0.3 = ",d2);
// Resultado:   c1/2+0.3 = 1.3

//--- segundo ejemplo
   d2=c1/2.0+0.3;
   Print("c1/2.0 + 0.3 = ",d2);
// Resultado:   c1/2.0+0.3 = 1.8
```

La expresión calculada se compone de dos partes. En el primer ejemplo la variable c1 del tipo char se transforma en la variable temporal del tipo int, puesto que el segundo operando en la operación de división, la constante 2, tiene el tipo int mayor. Como resultado de la división de números enteros 3/2 obtenemos el valor 1 que tiene el tipo int.

En la segunda operación del primer ejemplo el segundo operando es la constante 0.3 que tiene el tipo double, por eso el resultado de la primera operación se transforma en la variable temporal del tipo double con el valor 1.0.

En el segundo ejemplo la variable c1 del tipo char se transforma en la variable temporal del tipo double, puesto que el segundo operando en la operación de división, la constante 2.0, tiene el tipo double. Las siguientes transformaciones no se realizan.

La conversión de tipos numéricos

En las expresiones del lenguaje MQL5 se puede utilizar tanto la conversión explícita de tipos, como la implícita. La conversión explícita de tipos se escribe de la siguiente manera:

```
var_1 = (tipo)var_2;
```

En calidad de la variable var_2 puede ser una expresión o resultado de ejecución de la función. Se admite también la inscipción funcional de la conversión explícita de tipos:

```
var_1 = tipo(var_2);
```

© 2000-2025, MetaQuotes Ltd.

Vamos a analizar la conversión explícita a base del primer ejemplo.

```
//--- tercer ejemplo
   double d2=(double)c1/2+0.3;
   Print("(double)c1/2 + 0.3 = ",d2);
// Resultado:   (double)c1/2+0.3 = 1.80000000
```

Antes de realizar la operación de división la variable c1 se convierte al tipo double. Ahora la constante de número entero 2 se convierte al valor 2.0 del tipo double, puesto que en el resultado de la transformación el primer operando ha obtenido el tipo double. Prácticamente la conversión explícita es una operación unaria.

Además, al intentar realizar la conversión de tipos, el resultado puede salir de los márgenes del rango permitido. En este caso habrá el acortamiento. Por ejemplo:

```
   char c;
   uchar u;
   c=400;
   u=400;
   Print("c = ",c); // resultado c = -112
   Print("u = ",u); // resultado u = 144
```

Antes de ejecutar las operaciones (salvo la de asignación), ocurre la transformación en el tipo que tenga la mayor prioridad, y antes de las operaciones de asignación - en el tipo objetivo.

Ejemplos:

```
   int    i=1/2;         // no hay conversión de tipos, resultado: 0
   Print("i = 1/2  ",i);

   int k=1/2.0;          // la expresión se convierte al tipo double,
   Print("k=1/2  ",k);   // luego se convierte al tipo objetivo int, resultado: 0

   double d=1.0/2.0;     // no hay conversión de tipos, resultado: 0.5
   Print("d=1/2.0; ",d);

   double e=1/2.0;       // la expresión se convierte al tipo double,
   Print("e=1/2.0; ",e);// que coincide con el tipo objetivo, resultado: 0.5

   double x=1/2;         // la expresión del tipo int se convierte al tipo objetivo dou
   Print("x=1/2; ",x);   // resultado: 0.0
```

Durante la conversión del tipo long/ulong al tipo double puede que se pierda la precisión: si el valor entero es más de 9223372036854774784 o menos de -9223372036854774784.

```
void OnStart()
  {
   long l_max=LONG_MAX;
   long l_min=LONG_MIN+1;
//--- buscamos el valor máximo entero que no pierde la precisión cuando se convierte a
   while(l_max!=long((double)l_max))
      l_max--;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- buscamos el valor mínimo entero que no pierde la precisión cuando se convierte a
   while(l_min!=long((double)l_min))
      l_min++;
//--- ahora derivamos el intervalo encontrado para los números enteros
   PrintFormat("Si un número entero se convierte a double, tiene que "
               "encontrarse en el intervalo [%I64d, %I64d]",l_min,l_max);
//--- ahora vamos a ver qué es lo que pasa si el número se queda fuera de este interva
   PrintFormat("l_max+1=%I64d, double(l_max+1)=%.f, ulong(double(l_max+1))=%I64d",
               l_max+1,double(l_max+1),long(double(l_max+1)));
   PrintFormat("l_min-1=%I64d, double(l_min-1)=%.f, ulong(double(l_min-1))=%I64d",
               l_min-1,double(l_min-1),long(double(l_min-1)));
//--- el resultado será el siguiente
// Cuando un número entero se convierte a double, este número tiene que encontrarse de
// l_max+1=9223372036854774785, double(l_max+1)=9223372036854774800, ulong(double(l_ma
// l_min-1=-9223372036854774785, double(l_min-1)=-9223372036854774800, ulong(double(l_
  }
```

Las conversiones para el tipo string

El tipo string tiene la mayor prioridad entre los tipos simples. Por esta razón, si en la operación uno de los operandos tiene el tipo string, entonces el otro operando va a ser convertido al tipo string de una manera automática. Hay que tener en cuenta que para el tipo string se permite sólo operación binaria de suma. Está permitida la conversión explícita de la variable del tipo string a cualquier tipo numérico.

Ejemplos:

```
   string s1=1.0/8;            // la expresión se convierte al tipo double,
   Print("s1=1.0/8; ",s1);     // luego al tipo objetivo string,
// resultado:"0.12500000"(cadena de 10 símbolos)

   string s2=NULL;             // deinicialización de la cadena
   Print("s2=NULL; ",s2);      // resultado: cadena vacía
   string s3="Ticket N"+12345; // la expresión se convierte al tipo string
   Print("s3=\"Ticket N\"+12345",s3);

   string str1="true";
   string str2="0,255,0";
   string str3="2009.06.01";
   string str4="1.2345e2";
   Print(bool(str1));
   Print(color(str2));
   Print(datetime(str3));
   Print(double(str4));
```

La conversión de tipos de punteros de las clases bases a los punteros de las clases derivadas

© 2000-2025, MetaQuotes Ltd.

Los objetos de la clase derivada abiertamente también pueden considerarse como los objetos de la clase base que le corresponde. Esto lleva a algunos efectos interesantes. Por ejemplo, a pesar del hecho que los objetos de diferentes clases derivadas de una clase base, puedan diferenciarse de una manera significativa uno del otro, podemos crear su lista asociada (List) puesto que los consideramos como los objetos de la clase base. Pero lo invertido no es correcto: resulta que los objetos de la clase base no son los objetos de la clase derivada de una forma automática.

Se puede utilizar la conversión explícita de tipos para la transformación de los punteros de la clase base en los punteros de la clase derivada. Pero hay que estar completamente seguro de la admisibilidad de esta transformación, porque en el caso contrario surgirá un error crítico de tiempo de ejecución y el programa-mql5 se detendrá.

Conversión dinámica de tipos con ayuda del operador dynamic_cast

Existe la posibilidad de realizar la conversión dinámica de tipos con la ayuda del operador dynamic_cast, que puede ser aplicado solo a los punteros de clase. En este caso, la comprobación de la corrección de los tipos se realiza en el momento de ejecución del programa. Esto significa que al usar el operador dynamic_cast, el compilador no realiza la comprobación del tipo de datos usado para la conversión. En caso de que se realice la transformación de un puntero en un tipo de datos que no sea un tipo concreto de objeto, el resultado será el valor NULL.

```
dynamic_cast <type-id> ( expression )
```

El parámetro *type-id* entre corchetes angulares debe ser el puntero al tipo de clase anteriormente determinado. El tipo de operando *expression* (a diferencia de C++) puede ser cualquiera, excepto void.

Ejemplo:

```
class CBar { };
class CFoo : public CBar { };

void OnStart()
  {
   CBar bar;
//--- la conversión dinámica del tipo de puntero *bar al puntero *foo está permitida
   CFoo *foo = dynamic_cast<CFoo *>(&bar); // no aparecerá un error crítico de ejecuci
   Print(foo);                             // foo=NULL
//--- el intento de conversión explícita del enlace de un objeto del tipo Bar a un obj
   foo=(CFoo *)&bar;                       // aparecerá un error crítico de ejecución
   Print(foo);                             // esta línea no se ejecutará
  }
```

Véase también

Tipos de datos

© 2000-2025, MetaQuotes Ltd.

Tipo void y constante NULL

Sintácticamente el tipo void es un tipo fundamental igual que el tipo char, uchar, bool, short, ushort, int, uint, color, long, ulong, datetime, float, double y string. Este tipo se utiliza o para indicar que la función no devuelve los valores, o como parámetro de la función que significa la falta de los parámetros.

La variable constante predeterminada **NULL** tiene el tipo *void*. Ésta puede ser asignada sin transformación a las variables de cualquier otro tipo fundamental. También se permite la comparación de las variables de tipos fundamentales con el valor **NULL**

Ejemplo:

```
//--- si la cadena no está inicializada, entonces le asignaremos nuestro valor predete
if(some_string==NULL) some_string="empty";
```

Además, **NULL** se puede comparar con los punteros a los objetos creados con el operador new.

Véase también

Variables, Funciones

© 2000-2025, MetaQuotes Ltd.

Tipos personalizados

La palabra clave typedef en el lenguaje C++ permite crear tipos de datos personalizados, para ello, basta con definir el nuevo nombre del tipo de datos para un tipo de datos ya existente. Además, en este caso, el propio tipo de datos no se crea: solo se define el nuevo nombre para un tipo de datos ya existente. Gracias al uso de tipos personalizados es posible hacer los programas más flexibles: para ello, a veces basta con cambiar las instrucciones typedef con la ayuda de macro-sustituciones (#define). El uso de tipos personalizados permite mejorar la legibilidad del código, ya que para los tipos de datos estándar, con la ayuda de typedef se pueden usar nombres descriptivos propios. El formato general de registro de instrucciones para crear un tipo personalizado es el siguiente:

```
   typedef tipo nuevo_nombre;
```

Aquí el elemento *tipo* designa cualquier tipo de datos permisible, y el elemento *nuevo_nombre*, el nuevo nombre para este tipo. Es importante destacar que el nuevo nombre se define solo como adición a un nombre de tipo ya existente, pero no lo sustituye. En el lenguaje MQL5, con la ayuda de typedef, se pueden crear punteros a una función.

Puntero a una función

En general, el puntero a una función se define con el formato de registro

```
   typedef tipo_de_resultado_de_la_función (*Nombre_del_tipo_de_función)(lista_de_tipo
```

donde, después de la palabra typedef se establece la signatura de la función, es decir, el número y el tipo de parámetros de entrada, así como el tipo de resultado retornado por la función. Como explicación, mostraremos un sencillo ejemplo de creación y uso de un puntero a una función:

```
//--- declaramos el puntero a una función, que usa dos parámetros del tipo int
   typedef int (*TFunc)(int,int);
//--- TFunc es el tipo, podemos declarar la variable-puntero a una función
   TFunc func_ptr; // puntero a una función
//--- declaramos las funciones que corresponden a la descripción de TFunc
   int sub(int x,int y) { return(x-y); }  // resta de una cifra a la otra
   int add(int x,int y) { return(x+y); }  // suma de una cifra a la otra
   int neg(int x)       { return(~x);  }  // invertir bits en la variable
//--- en la variable func_ptr se puede guardar la dirección de la función, para llamar
   func_ptr=sub;
   Print(func_ptr(10,5));
   func_ptr=add;
   Print(func_ptr(10,5));
   func_ptr=neg;           // error: neg no tiene el tipo  int (int,int)
   Print(func_ptr(10));    // error: debe haber dos parámetros
```

En este ejemplo, a la variable *func_ptr* se le pueden asignar las funciones *sub* y *add*, puesto que cada una de ellas tiene dos parámetros int, como se indica en la definición del puntero a la función *TFunc*. A su vez, la función *neg* no puede ser asignada al puntero func_ptr, puesto que su signatura se diferencia.

Organización de los modelos de evento en la interfaz de usuario

© 2000-2025, MetaQuotes Ltd.

Con la ayuda de los punteros a funciones es sencillo construir el procesamiento de eventos a la hora de crear una interfaz de usuario. Usando como ejemplo el apartado CButton, mostraremos cómo crear botones y añadirles funciones para procesar la pulsación. Primero tenemos que definir el puntero a la función *TAction*, que se llamará al pulsar un botón, y después crear tres funciones de acuerdo con la descripción de *TAction*.

```
//--- creamos el tipo de función personalizado
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Abrir archivo                                                   |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("Se ha llamado la función %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Guardar archivo                                                 |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("Se ha llamado la función %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Cerrar archivo                                                  |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("Se ha llamado la función %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
```

Después reproduciomos la clase MyButton de CButton, en la que añadimos el miembro *TAction*, que es el puntero a la función.

```
//+------------------------------------------------------------------+
//| Creando nuestra propia clase de botón con la función de procesamiento de eventos
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // procesador de eventos del gráfic
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- constructor con indicación de texto del botón y el puntero a la función para
                     MyButton(string text, TAction act)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Text(text);
      m_action=act;
     }
   //--- instalando la función propia, que se llamará desde el procesador de eventos C
   void              SetAction(TAction act){m_action=act;}
   //--- procesador estándar de eventos del gráfico
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL && lparam==Id())
        {
         //--- llamamos el procesador propio m_action()
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- retornamos el  resultado de la llamada del procesador desde la clase padre
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
```

A continuación, creamos la clase derivada CControlsDialog de CAppDialog, en la que añadimos la matriz *m_buttons* para guardar los botones del tipo *MyButton*, así como los métodos *AddButton(MyButton &button)* y *CreateButtons()*.

```
//+------------------------------------------------------------------+
//| Clase CControlsDialog                                            |
//| Cometido: panel gráfico para controlar la aplicación             |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // matriz de botones
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- añadir botón
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- crear botón
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| Crear objeto CControlsDialog en el gráfico                       |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
   return(CreateButtons());
//---
  }
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Crear y añadir botones en el panel CControlsDialog               |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
//--- calculando las coordenadas de los botones
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- añadimos los objetos de los botones junto con los punteros a las funciones
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",::Save));
   AddButton(new MyButton("Close",Close));
//--- creamos los botones gráficamente
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("Failed to create button %s %d",b.Text(),i);
         return(false);
        }
      //--- añadimos cada botón al contenedor CControlsDialog
      if(!Add(b))
         return(false);
     }
//--- succeed
   return(true);
  }
```

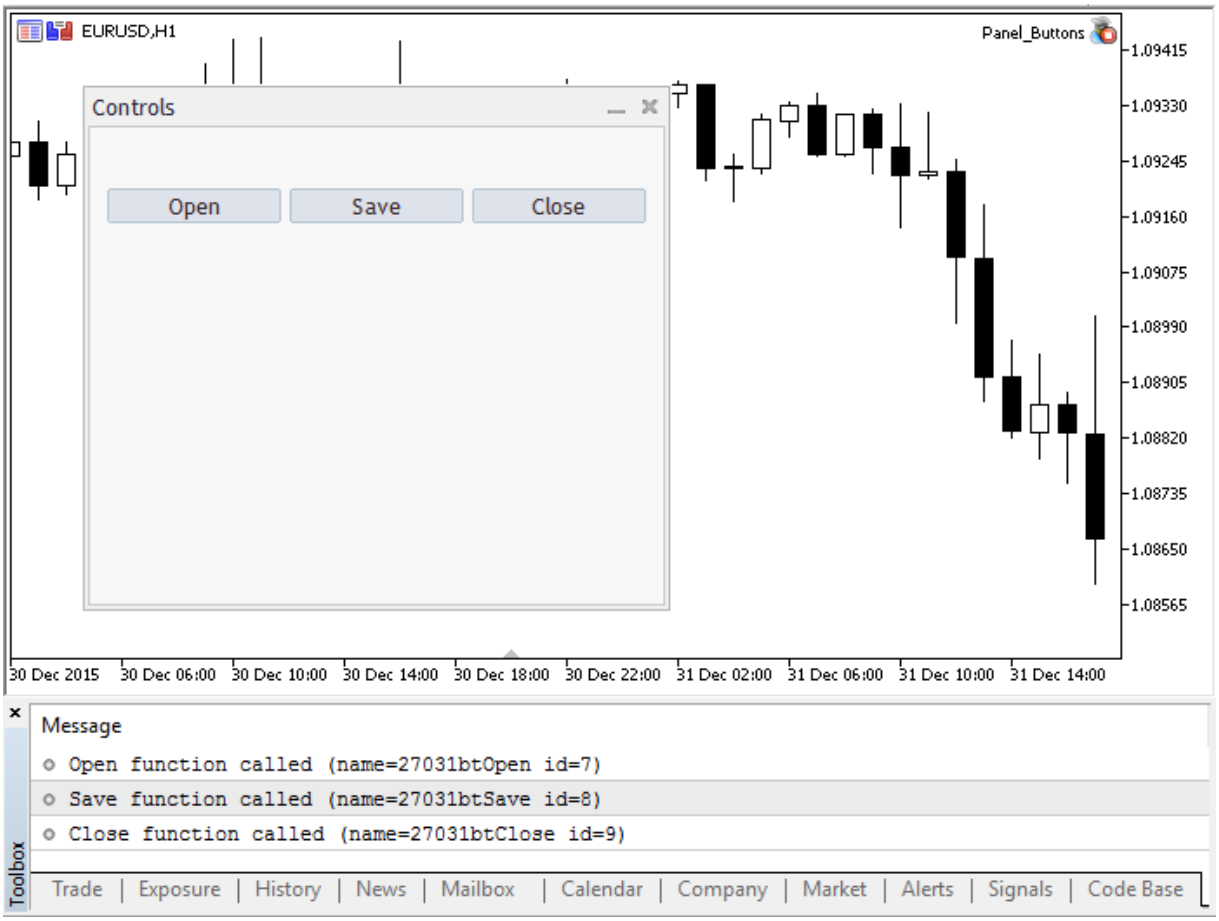
© 2000-2025, MetaQuotes Ltd.

Ahora podemos escribir un programa con el uso del panel de control CControlsDialog, en el que se crean 3 botones "Open", "Save" y "Close". Al pulsar un botón, se llama la función que le corresponde, escrita en forma de puntero a la función *TAction*.

```
//--- declaramos el objeto a nivel global, para crearlo de forma automática al iniciar
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ahora creamos el objeto en el gráfico
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- iniciamos la aplicación
   MyDialog.Run();
//--- la aplicación ha sido iniciada con éxito
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destroy dialog
   MyDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
//--- para los eventos del gráfico llamamos el procesador desde la clase padre (CAppDi
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

En la imagen se muestra el aspecto externo de la aplicación iniciada, así como los resultados de la pulsación de los botones.

© 2000-2025, MetaQuotes Ltd.



Código fuente completo del programa

```
//+------------------------------------------------------------------+
//|                                                Panel_Buttons.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+

#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panel con varios botones CButton"
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
```

© 2000-2025, MetaQuotes Ltd.

```
//--- for buttons
#define BUTTON_WIDTH                       (100)      // size by X coordinate
#define BUTTON_HEIGHT                      (20)       // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                        (20)       // size by Y coordinate

//--- creando el tipo personalizado de función
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Abrir archivo                                                   |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("Se ha llamado la función %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Guardar archivo                                                 |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("Se ha llamado la función %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Cerrar archivo                                                  |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("Se ha llamado la función %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
//+------------------------------------------------------------------+
//| Creando nuestra propia clase de botón con la función de procesamiento de eventos
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // procesador de eventos del gráfic
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- constructor con indicación de texto del botón y el puntero a la función para
                     MyButton(string text,TAction act)
     {
      Text(text);
      m_action=act;
     }
   //--- instalando la función propia, que se llamará desde el procesador de eventos C
```

© 2000-2025, MetaQuotes Ltd.

```
   void              SetAction(TAction act){m_action=act;}
   //--- procesador estándar de eventos del gráfico
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL && lparam==Id())
        {
         //--- llamamos el procesador propio
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- retornamos el  resultado de la llamada del procesador desde la clase padre
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
//+------------------------------------------------------------------+
//| Clase CControlsDialog                                            |
//| Cometido: panel gráfico para controlar la aplicación       |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // matriz de botones
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- añadir botón
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- crear botón
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| Crear objeto CControlsDialog en el gráfico                       |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
   return(CreateButtons());
//---
  }
//+------------------------------------------------------------------+
//| Crear y añadir botones en el panel CControlsDialog           |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- calculando las coordenadas de los botones
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- añadimos los objetos de los botones junto con los punteros a las funciones
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",::Save));
   AddButton(new MyButton("Close",Close));
//--- creamos los botones gráficamente
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("Failed to create button %s %d",b.Text(),i);
         return(false);
        }
      //--- añadimos cada botón al contenedor CControlsDialog
      if(!Add(b))
         return(false);
     }
//--- succeed
   return(true);
  }
//--- declaramos el objeto a nivel global, para crearlo de forma automática al iniciar
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ahora creamos el objeto en el gráfico
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- iniciamos la aplicación
   MyDialog.Run();
//--- la aplicación ha sido iniciada con éxito
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destroy dialog
   MyDialog.Destroy(reason);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
//--- para los eventos del gráfico llamamos el procesador desde la clase padre (CAppDi
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Vea también

Variables, funciones

© 2000-2025, MetaQuotes Ltd.

Punteros a objetos

En MQL5, es posible crear dinámicamente objetos de tipo complejo. Esto se hace usando el operador new,que retorna el descriptor del objeto creado. El descriptor tiene un tamaño de 8 bytes. Sintácticamente, los descriptores de los objetos en MQL5 son similares a los punteros en C++.

Ejemplo:

```
MyObject* hobject= new MyObject();
```

A diferencia de C++, la variable hobject en el ejemplo anterior no es un puntero de memoria, sino un descriptor de objeto. Además, en MQL5, todos los objetos de los parámetros de las funciones deben transmitirse por referencia. Ejemplos de transmisión de objetos como parámetros de funciones:

```
class Foo
  {
public::
   string            m_name;
   int               m_id;
   static int        s_counter;
   //--- constructores y destructores
                     Foo(void){Setup("noname");};
                     Foo(string name){Setup(name);};
                    ~Foo(void){};
   //--- inicializamos el objeto Foo
   void              Setup(string name)
     {
      m_name=name;
      s_counter++;
      m_id=s_counter;
     }
  };
int Foo::s_counter=0;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos el objeto como una variable con creación automática
   Foo foo1;
//--- variante de transmisión de un objeto por referencia
   PrintObject(foo1);

//--- declaramos un puntero a un objeto y lo creamos con el operador 'new'
   Foo *foo2=new Foo("foo2");
//--- variante de transmisión de un puntero a un objeto por referencia
   PrintObject(foo2); // el puntero a un objeto es convertido automáticamente por el c

//--- declaramos un array de objetos Foo
   Foo foo_objects[5];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- variante de transmisión de un array de objetos
   PrintObjectsArray(foo_objects); // función separada para pasar un array de objetos

//--- declaramos un array de punteros a los objetos Foo
   Foo *foo_pointers[5];
   for(int i=0;i<5;i++)
      foo_pointers[i]=new Foo("foo_pointer");
//--- variante de transmisión de un array de objetos
   PrintPointersArray(foo_pointers); // función aparte para transmitir un array de obj

//--- antes de completar el trabajo, nos aseguramos de borrar los objetos creados como
   delete(foo2);
//--- borramos el array de punteros
   int size=ArraySize(foo_pointers);
   for(int i=0;i<5;i++)
      delete(foo_pointers[i]);
//---
  }
//+------------------------------------------------------------------+
//| Los objetos siempre se transmiten por referencia |
//+------------------------------------------------------------------+
void PrintObject(Foo &object)
  {
   Print(__FUNCTION__,": ",object.m_id," Object name=",object.m_name);
  }
//+------------------------------------------------------------------+
//| Transmisión de un array de objetos |
//+------------------------------------------------------------------+
void PrintObjectsArray(Foo &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
//| Transmisión de un array de punteros de objeto                    |
//+------------------------------------------------------------------+
void PrintPointersArray(Foo* &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
```

Comprobación del puntero antes de su uso

© 2000-2025, MetaQuotes Ltd.

El intenta de acceso a un puntero no válido provoca la finalización crítica del programa. Para comprobar un puntero antes de usarlo, utilice la función CheckPointer. El puntero puede no ser válido en los siguientes casos:

- el puntero es igual a NULL;
- el objeto ha sido destruido por el operador delete.

Esta función se puede usar como comprobación de la corrección de un puntero. Un valor distinto a cero asegura que se pueda acceder a los datos usando este puntero.

```
class CMyObject
 {
protected:
  double              m_value;
public::
                      CMyObject(void);
                      CMyObject(double value) {m_value=value;};
                     ~CMyObject(void){};
  //---
  double              Value(void) {return(m_value);}
 };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- creamos un objeto no inicializado
  CMyObject *pointer;
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

//--- inicializamos el puntero
  pointer=new CMyObject(M_PI);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("2. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=", pointer.Value());

//--- eliminamos el objeto
  delete(pointer);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("3. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=", pointer.Value());
 }
/*
   1. pointer is POINTER_INVALID
   2. pointer.Value()=3.141592653589793
```

© 2000-2025, MetaQuotes Ltd.

```
   3. pointer is POINTER_INVALID
*/
```

También podemos usar el operador "!" (para comprobar rápidamente el punteroLNOT), que comprueba su validez usando una llamada implícita a la función CheckPointer. Esto nos permite escribir el código de forma más concisa y clara. Este es el aspecto que tendrían las comprobaciones del ejemplo anterior:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- creamos un objeto no inicializado
  CMyObject *pointer;
  if(!pointer)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

//--- inicializamos el puntero
  pointer=new CMyObject(M_PI);
  if(!pointer)
    Print("2. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=", pointer.Value());

//--- eliminamos el objeto
  delete(pointer);
  if(!pointer)
    Print("3. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=", pointer.Value());
 }
/*
   1. pointer is POINTER_INVALID
   2. pointer.Value()=3.141592653589793
   3. pointer is POINTER_INVALID
*/
```

Para comprobar rápidamente la existencia de NULL, usamos el operador "==". Por ejemplo: ptr==NULL o ptr!=NULL.

Véase también

Variables, Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Referencias Modificador & y palabra clave this

Traspaso de parámetros por referencia

En MQL5 los parámetros de tipos simples pueden ser traspasados por valor y por referencia, mientras que los parámetros de tipos compuestos siempre se traspasan por referencia. Para indicar al compilador la necesidad de traspaso de parámetros por referencia antes del nombre del parámetro se coloca el signo **&**.

El traspaso de parámetro por referencia significa el paso de dirección de una variable, por eso todas las modificaciones realizadas con el parámetro pasado por referencia en seguida se reflejarán también en la variable original. Al usar el traspaso de parámetros por referencia, se puede organizar al mismo tiempo el retorno de varios resultados desde la función. Para evitar los cambios del parámetro traspasado por referencia, hace falta utilizar el modificador const.

De esa manera, si el parámetro entrante de una función es un array, objeto de estructura o clase, entonces en el encabezado de la función después del tipo de variable y antes de su nombre se pone el signo '&'.

Ejemplo

```
class CDemoClass
  {
private:
   double            m_array[];

public:
   void              setArray(double &array[]);
  };
//+------------------------------------------------------------------+
//| relleno del array                                                |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
     ArrayResize(m_array,ArraySize(array));
     ArrayCopy(m_array, array);
     }
  }
```

En el ejemplo de arriba se declara la clase CDemoClass que contiene un elemento-array privado m_array[] del tipo double. Se declara la función setArray() en la que el array array[] es traspasado por referencia. Si escribimos el encabezado de la función sin indicar el traspaso por referencia, es decir, quitando el signo de ampersand, durante el intento de compilación de tal código saldrá un mensaje de error.

A pesar de que un array se traspasa por referencia, no podemos realizar la asignación de un array a otro. Es necesario hacer una copia elemento por elemento del contenido de array-fuente al array-receptor. Para los arrays y estructuras a la hora de traspaso en calidad del parámetro de la función la presencia del signo & es obligatoria durante la descripción de la función.

Palabra clave this

© 2000-2025, MetaQuotes Ltd.

La variable del tipo clase (objeto) puede ser traspasada tanto por referencia, como por puntero. Un puntero, igual como una referencia, sirve para obtener el acceso a un objeto. Después de declarar el puntero a objeto es necesario aplicarle el operador new para su creación e inicialización.

La palabra reservada **this** sirve para obtener referencia del objeto a si mismo, dicha referencia debe ser accesible dentro de los métodos de la clase o estructura. **this** siempre hace referencia al objeto en el método del que se utiliza. Y la expresión GetPointer(this) da un puntero al objeto al que pertenece la función en la que se realiza la llamada a la función GetPointer(). En MQL5 las funciones no pueden devolver los objetos pero está permitido devolver el puntero a objetos.

Así, si hace falta que la función devuelva el objeto, podemos devolver el puntero a este objeto a modo de GetPointer(this). Añadamos a la descripción de la clase CDemoClass la función getDemoClass() que devuelve el puntero a objeto de esta clase.

```
class CDemoClass
  {
private:
   double            m_array[];

public:
   void              setArray(double &array[]);
   CDemoClass       *getDemoClass();
  };
//+------------------------------------------------------------------+
//| relleno del array                                                |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
      ArrayResize(m_array,ArraySize(array));
      ArrayCopy(m_array,array);
     }
  }
//+------------------------------------------------------------------+
//| devuelve su propio puntero                                       |
//+------------------------------------------------------------------+
CDemoClass *CDemoClass::getDemoClass(void)
  {
   return(GetPointer(this));
  }
```

Las estructuras no disponen de punteros, no se les puede aplicar los operadores *new* y *delete*, tampoco se puede usar GetPointer(this.

Véase también

Punteros a objetos, Creación y eliminación de objetos, Visibilidad y tiempo de vida de variables

© 2000-2025, MetaQuotes Ltd.

Operaciones y expresiones

Algunos caracteres y sucesiones de caracteres adquieren importancia especial. Son llamados símbolos de operaciones, por ejemplo:

```
+ - * / %     símbolos de operaciones aritméticas
&& ||         símbolos de operaciones lógicas
= += *=       símbolos de operaciones de asignación
```

Los símbolos de operaciones se utilizan en las expresiones y tienen sentido cuando se les dan los operandos correspondientes. Además, se da importancia especial a los signos de puntuación. Los signos de puntuación incluyen los paréntesis, llaves, coma, dos puntos y el punto coma.

Los símbolos de operaciones, signos de puntuación y espacios sirven para separar los elementos del lenguaje.

En este apartado se examinan los temas siguientes:

- Expresiones
- Operaciones aritméticas
- Operaciones de asignación
- Operaciones de relación
- Operaciones lógicas
- Operaciones a nivel de bits
- Otras operaciones
- Prioridades y orden de operaciones

© 2000-2025, MetaQuotes Ltd.

Expresiones

Una expresión se compone de uno o varios operandos y símbolos de operaciones. Se puede escribirla en varias líneas.

Ejemplos:

```
a++; b = 10;          // varias expresiones se ubican en una línea
//---  una expresión se divide en varias líneas
x = (y * z) /
    (w + 2) + 127;
```

La expresión terminada con el punto y coma (;) es un operador.

Véase también

Prioridades y orden de las operaciones

© 2000-2025, MetaQuotes Ltd.

Operaciones aritméticas

Las operaciones aditivas y multiplicativas pertenecen a las operaciones aritméticas:

```
Suma de valores                          i = j + 2;
Resta de valores                         i = j - 3;
Cambio de signo                          x = - x;
Multiplicación de valores                z = 3 * x;
Cociente de la división                  i = j / 5;
Resto de la división                     minutes = time % 60;
Suma de 1 al valor de la variable        i++;
Suma de 1 al valor de la variable        ++i;
Resta de 1 del valor de la variable      k--;
Resta de 1 del valor de la variable      --k;
```

Operaciones de incremento y decremento se aplican unicamente a las variables; no se usan con las constantes. El incremento prefijo (++i) y decremento prefijo (--k) se aplican a la variable justo antes de usarla en la expresión.

Post-incremento (i++) y post-decremento (k--) se aplican a la variable justo después de usarla en la expresión.

Observación importante

```
int i=5;
int k = i++ + ++i;
```

Pueden surgir problemas de cómputo a la hora de pasar la expresión arriba mencionada de un entorno de programación a otro (por ejemplo, de Borland C++ a MQL5). Generalmente el orden de cálculos depende de la implementación del compilador. En la práctica existen dos modos de implementar un post-decremento (post-incremento):

1. el post-decremento (post-incremento) se aplica a la variable tras calcular toda la expresión;
2. el post-decremento (post-incremento) se aplica a la variable directamente durante la operación.

En MQL5 en estos momentos está implementado el primer modo de calcular el post-decremento (post-incremento). Pero incluso teniendo este conocimiento, es mejor no experimentar con el uso de este detalle.

Ejemplos:

```
int a=3;
a++;             // expresión correcta
int b=(a++)*3;   // expresión incorrecta
```

Véase también

Prioridades y orden de las operaciones

© 2000-2025, MetaQuotes Ltd.

Operaciones de asignación

El valor de la expresión que incluye la operación de asignación, es el valor del operando izquierdo después de la asignación.

```
Asignación del valor x a la variable y                                  y = x;
```

Las siguientes operaciones unen las operaciones aritméticas y a nivel de bits con la operación de asignación:

```
Sumando x al valor de la variable y                                     y += x;
Restando x del valor de la variable y                                   y -= x;
Multiplicando el valor de la variable y por x                           y *= x;
Dividiendo el valor de la variable y por x                              y /= x;
Resto de la división del valor de la variable y por x                   y %= x;
Desplaza x bits a la derecha la representación binaria de y             y >>= x;
Desplaza x bits a la izquierda la representacion binaria de y           y <<= x;
Operación a nivel de bits AND de las representaciones binarias y y x    y &= x;
Operación a nivel de bits OR de las representaciones binarias y y x     y |= x;
Operación a nivel de bits excluyendo OR
          de las representaciones binarias y y x                        y ^= x;
```

Operaciones a nivel de bits se realizan unicamente con números enteros. Durante la operación del desplazamiento lógico de la representación y a la derecha/izquierda sobre x bits, se usan 5 menores dígitos binarios del valor x, los dígitos mayores se omiten; es decir, el desplazamiento se realiza sobre 0-31 bits.

Durante la operación %= (valor y por el módulo x) el signo del resultado coincide con el signo del dividiendo.

El operador de asignación puede aparecer varias veces en una expresión. En este caso el procesamiento de la expresión se realiza de derecha a izquierda:

```
y=x=3;
```

Al principio a la variable x se le asigna el valor 3, luego a la variable y se le va a asignar el valor de la variable x, es decir, también 3.

Véase también

Prioridades y orden de las operaciones

© 2000-2025, MetaQuotes Ltd.

Operaciones de relación

El valor lógico FALSO se representa por el valor entero cero, y el valor VERDADERO se representa por cualquier valor distinto a cero.

FALSO (0) y VERDADERO (1) son valores de las expresiones que contienen las operaciones de relación o las operaciones lógicas.

```
Verdadero, si a es igual a b                    a == b;
Verdadero, si a no es igual a b                 a != b;
Verdadero, si a es menor que b                   a < b;
Verdadero, si a es mayor que b                   a > b;
Verdadero, si a es menor o igual a b          a <= b;
Verdadero, si a es mayor o igual a b          a >= b;
```

No se puede comparar la equivalencia de dos números reales. En mayoría de los casos resulta que dos números aparentemente iguales pueden ser desiguales por la diferencia del valor en el décimoquinto dígito después de la coma. Para una comparación correcta de dos números reales es necesario comparar la diferencia normalizada de estos números con el valor cero.

Ejemplo:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0) return(true);
   else return(false);
  }
void OnStart()
  {
   double first=0.3;
   double second=3.0;
   double third=second-2.7;
   if(first!=third)
     {
      if(CompareDoubles(first,third))
         printf("%.16f  %.16f   son iguales",first,third);
     }
  }
// Resultado: 0.3000000000000000  0.2999999999999998   no obstante, son iguales
```

Véase también

Prioridades y orden de las operaciones

© 2000-2025, MetaQuotes Ltd.

Operaciones lógicas

Negación lógica NOT(!)

El operando de la operación de negación lógica NOT(!) debe tener el tipo aritmético. El resultado es igual a VERDADERO(1) si el valor del operando es FALSO(0), y es igual a FALSO(0) si el operando no es igual a FALSO(0).

```
if(!a) Print("no 'a'");
```

Operación lógica OR (||)

Operación lógica OR (||) de los valores x y y. El valor de la expresión es VERDADERO(1) si el valor de x o y es verdadero (no cero). En caso contrario es FALSO(0).

```
if(x<0 || x>=max_bars) Print("out of range");
```

Operación lógica AND (&&)

Operación lógica AND (&&) de los valores x y y. El valor de la expresión es VERDADERO(1) si el valor de x y y son verdaderos (no cero). En caso contrario es FALSO(0).

Breve estimación de operaciones lógicas

Tratándose de operaciones lógicas, se les aplica un esquema llamado "de breve estimación". Esto quiere decir que el cálculo de una operación lógica se finaliza cuando su resultado puede ser estimado de forma precisa.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- el primer ejemplo de breve estimación
   if(func_false() && func_true())
     {
      Print("Operación &&: Este mensaje nunca estará visible para Ud.");
     }
   else
     {
      Print("Operación &&: El resultado de la primera expresión es false, por eso la s
     }
//--- el segundo ejemplo de breve estimación
   if(!func_false() || !func_true())
     {
      Print("Operación ||: El resultado de la primera expresión es true, por eso la se
     }
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("Operación ||: Este mensaje nunca estará visible para Ud.");
      }
  }
//+------------------------------------------------------------------+
//| la función siempre devuelve false                                |
//+------------------------------------------------------------------+
bool func_false()
  {
   Print("Función func_false()");
   return(false);
  }
//+------------------------------------------------------------------+
//| la función siempre devuelve true                                 |
//+------------------------------------------------------------------+
bool func_true()
  {
   Print("Función func_true()");
   return(true);
  }
```

Véase también

Prioridades y orden de las operaciones

© 2000-2025, MetaQuotes Ltd.

Operaciones a nivel de bits

Complemento a uno

Complemento del valor de la variable a uno. El valor de la expresión contiene 1 en todos los dígitos donde el valor de la variable contiene 0, y 0 en todos los dígitos donde el valor de la variable contiene 1.

```
b = ~n;
```

Ejemplo:

```
   char a='a',b;
   b=~a;
   Print("a = ",a, "  b = ",b);
// Resultado va a ser el siguiente::
// a = 97   b = -98
```

Desplazamiento a la derecha

La representación binaria de x se desplaza a la derecha a y dígitos. Si el valor desplazado posee el tipo sin signos, entonces se realiza el desplazamiento lógico a la derecha, es decir, los dígitos liberados por la izquierda se rellenan con ceros.

Pero si el valor desplazado tiene el tipo de signo, entonces se realiza el desplazamiento aritmético a la derecha, es decir, los dígitos liberados por la izquierda van a ser rellenados con el valor del bit de signo (si el número es positivo, el valor del bit de signo es igual a 0; si el número es negativo, el valor del bit de signo es igual a 1)

```
x = x >> y;
```

Ejemplo:

```
   char a='a',b='b';
   Print("Before:  a = ",a, "  b = ",b);
//--- hagamos el desplazamiento a la derecha
   b=a >> 1;
   Print("After:   a = ",a, "  b = ",b);
// Resultado va a ser el siguiente:
// Before:  a = 97   b = 98
// After:   a = 97   b = 48
```

Desplazamiento a la izquierda

La representación binaria de x se desplaza a la izquierda a y dígitos, rellenándose los dígitos liberados por la derecha con ceros.

```
x = x << y;
```

Ejemplo:

```
   char a='a',b='b';
   Print("Before:  a = ",a, "  b = ",b);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- hagamos el desplazamiento a la izquierda
   b=a << 1;
   Print("After:   a = ",a, "  b=",b);
// Resultado va a ser el siguiente:
// Before:  a = 97   b = 98
// After:   a = 97   b = -62
```

No se recomienda realizar el desplazamiento a más o igual cantidad de bits que la longitud de la variable desplazada, puesto que el resultado de dicha operación no está definido.

Operación a nivel de bits AND

Operación a nivel de bits AND de las representaciones binarias y y x. El valor de la expresión contiene 1 (VERDADERO) en todos los dígitos en los que como x, tanto y no contienen ceros; y 0 (FALSO) en los demás dígitos.

```
b = ((x & y) != 0);
```

Ejemplo:

```
   char a='a',b='b';
//--- operación AND
   char c=a&b;
   Print("a = ",a,"  b = ",b);
   Print("a & b = ",c);
// Resultado va a ser el siguiente:
// a = 97   b = 98
// a & b = 96
```

Operación a nivel de bits OR

Operación a nivel de bits OR de las representaciones binarias y y x. El valor de la expresión contiene 1 en todos los dígitos en los que x o y no contienen 0; y 0 en todos los demás dígitos.

```
b = x | y;
```

Ejemplo:

```
   char a='a',b='b';
//--- operación OR
   char c=a|b;
   Print("a = ",a,"  b = ",b);
   Print("a | b = ",c);
// Resultado va a ser el siguiente:
// a = 97   b = 98
// a | b = 99
```

Operación a nivel de bits excluyendo OR

Operación a nivel de bit excluyendo AND (eXclusive OR) de las representaciones binarias y y x. El valor de la expresión contiene 1 en los dígitos en los que x y y poseen los valores binarios diferentes; y 0 en todos los demás dígitos.

© 2000-2025, MetaQuotes Ltd.

```
b = x ^ y;
```

Ejemplo:

```
   char a='a', b='b';
//--- operación excluyendo OR
   char c=a^b;
   Print("a = ",a,"  b = ",b);
   Print("a ^ b = ",c);
// Resultado va a ser el siguiente:
// a = 97   b = 98
// a ^ b = 3
```

Operaciones a nivel de bits se realizan unicamente con los números enteros.

Véase también

Prioridades y orden de las operaciones

© 2000-2025, MetaQuotes Ltd.

Otras operaciones

Indexación ([])

A la hora de dirigirse al elemento número i del array, el valor de la expresión será el valor de la variable con el número i.

Ejemplo:

```
array[i] = 3; // Adjudicar el valor 3 al elemento número i del array array.
```

Sólo el número entero puede ser un índice del array. Se admiten sólo los arrays cuatro-dimensionales. La indexación de cada medición se realiza desde 0 hasta **el tamaño de medición**-1. En un caso particular de un array unidimensional compuesto por 50 elementos, la referencia hacia el primer elemento se verá como el array[0], hacia el último elemento - array[49].

Al acceder fuera de los límites del array, el subsistema ejecutivo generará un error crítico y la ejecución del programa se detendrá.

Llamada a función con los argumentos x1, x2,..., xn

Cada argumento puede representar una constante, variable o expresión del tipo correspondiente. Los argumentos traspasados se separan con comas y deben encerrarse entre los paréntesis, el paréntesis que abre debe seguir detrás del nombre de la función que se llama.

El valor de la función es el valor que se devuelve a la función. Si el tipo del valor devuelto a la función es void, no se puede colocar la llamada a esta función en la parte derecha dentro de la operación de asignación. Preste atención a que el orden de ejecución de las expresiones x1,..., xn se garantiza.

Ejemplo:

```
   int length=1000000;
   string a="a",b="b",c;
//---
   int start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);
```

Operación coma (,)

Las expresiones separadas con coma se calculan de izquierda a derecha. Todos los efectos secundarios derivados del cálculo de la expresión izquierda pueden surgir antes de calcular la expresión de la

© 2000-2025, MetaQuotes Ltd.

derecha. El tipo y el valor del resultado coinciden con el tipo y el valor de la expresión derecha. Como ejemplo podemos estudiar la lista de los parámetros traspasados (véase más arriba).

Ejemplo:

```
for(i=0,j=99; i<100; i++,j--) Print(array[i][j]);
```

Operación punto (.)

Para el acceso directo a los elementos públicos de las estructuras y clases se utiliza la operación punto. Sintaxis

```
Nombre_de_la_variable_tipo_estructura.Nombre_del_elemento
```

Ejemplo:

```
   struct SessionTime
     {
      string sessionName;
      int    startHour;
      int    startMinutes;
      int    endHour;
      int    endMinutes;
     } st;
   st.sessionName="Asian";
   st.startHour=0;
   st.startMinutes=0;
   st.endHour=9;
   st.endMinutes=0;
```

Operación de resolución de contexto (::)

En el programa mql5 cada función tiene su contexto de ejecución. Por ejemplo, la función del sistema Print() se ejecuta en el contexto global. Las funciones importadas se llaman en el contexto de importación correspondiente. Las funciones-métodos de las clases tienen el contexto de la clase correspondiente. La sintaxis de la operación de resolución de contexto:

```
[Nombre_del_contexto]::Nombre_de_función(parámetros)
```

Si falta el nombre del contexto, es una indicación directa a la utilización del contexto global. En caso de ausencia de la operación de resolución de contexto la función se busca en el contexto más próximo. Si en el contexto local no hay ninguna función, la busqueda se realiza en el contexto global.

Además, la operación de resolución de contexto se usa para definir la función-miembro de la clase.

```
tipo Nombre_de_clase::Nombre_de_función(descripción_de_parámetros)
   {
// cuerpo de la función
   }
```

© 2000-2025, MetaQuotes Ltd.

Use of several functions of the same name from different execution contexts in a program may cause ambiguity. The priority order of function calls without explicit scope specification is the following:

1. Class methods. If no function with the specified name is set in the class, move to the next level.
2. MQL5 functions. If the language does not have such a function, move to the next level.
3. User defined global functions. If no function with the specified name is found, move to the next level.
4. Imported functions. If no function with the specified name is found, the compiler returns an error.

To avoid the ambiguity of function calls, always explicitly specify the function scope using the scope resolution operation.

Ejemplo:

```
#property script_show_inputs
#import "kernel32.dll"
   int GetLastError(void);
#import

class CCheckContext
  {
   int          m_id;
public:
                CCheckContext() { m_id=1234; }
protected:
   int          GetLastError() { return(m_id); }
  };
class CCheckContext2 : public CCheckContext
  {
   int          m_id2;
public:
                CCheckContext2() { m_id2=5678; }
   void         Print();
protected:
   int          GetLastError() { return(m_id2); }
  };
void CCheckContext2::Print()
  {
   ::Print("Terminal GetLastError",::GetLastError());
   ::Print("kernel32 GetLastError",kernel32::GetLastError());
   ::Print("parent GetLastError",CCheckContext::GetLastError());
   ::Print("our GetLastError",GetLastError());
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CCheckContext2 test;
   test.Print();
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

Operación de obtención del tamaño de tipos de datos o el tamaño de objeto de cualquier tipo de datos (sizeof)

Utilizando la operación sizeof se puede determinar el tamaño de la memoria que corresponde al identificador o al tipo. La operación sizeof tiene el formato siguiente:

Ejemplo:

```
sizeof(expresión)
```

Cualquier identificador o el nombre del tipo encerrado entre paréntesis puede ser utilizado como una expresión. Cabe mencionar que no se puede usar el nombre del tipo void, y el identificador no puede pertenecer al campo de bits, o ser el nombre de la función.

Si la expresión es el nombre del array estático (es decir, se da la primera dimensión), entonces el resultado es el tamaño de todo el array (es decir, producto de multiplicación del número de elementos por longitud del tipo). Si la expresión es el nombre del array dinámico (no se da la primera dimensión), entonces el resultado es el tamaño del objeto del array dinámico.

Cuando sizeof se aplica al nombre del tipo de la estructura o clase, o al identificador que posee el tipo de estructura o clase, el resultado es el tamaño real de la estructura o clase.

Ejemplo:

```
struct myStruct
  {
   char   h;
   int    b;
   double f;
  } str;
Print("sizeof(str) = ",sizeof(str));
Print("sizeof(myStruct) = ",sizeof(myStruct));
```

El cálculo del tamaño se realiza durante la fase de compilación.

Véase también

Prioridades y orden de las operaciones

© 2000-2025, MetaQuotes Ltd.

Prioridades y orden de las operaciones

Para cada grupo de operaciones de la tabla la prioridad es la misma. Cuanto más alta sea la prioridad del grupo, más arriba se encuentra éste en la tabla. El orden de ejecución determina la agrupación de operaciones y operandos.

Atención: La prioridad de ejecución de las operaciones en el lenguaje MQL5 corresponde a la prioridad adoptada en C++ y se diferencia de la prioridad aceptada en el lenguaje MQL4.

Operación	Descripción	Orden de ejecución
() [] .	Llamada a la función Selección del elemento del array Selección del elemento de la estructura	De izquierda a derecha
! ~ - ++ -- (tipo) sizeof	Negación lógica Negación a nivel de bits (complement) Cambio de signo Incremento a uno (increment) Decremento a uno (decrement) Conversión de tipo Cálculo del tamaño en bytes	De derecha a izquierda
* / %	Multiplicación División División inexacta	De izquierda a derecha
+ -	Suma Resta	De izquierda a derecha
<< >>	Desplazamiento a la izquierda Desplazamiento a la derecha	De izquierda a derecha
< <= > >=	Menos que Menos o igual Más que Más o igual	De izquierda a derecha
== !=	Igual a No es igual a	De izquierda a derecha
&	Operación a nivel de bits AND	De izquierda a derecha
^	Operación a nivel de bits excluyendo OR (eXclude OR)	De izquierda a derecha
|	Operación a nivel de bits OR	De izquierda a derecha
&&	Operación lógica AND	De izquierda a derecha
||	Operación lógica OR	De izquierda a derecha
?:	Operación condicional	De derecha a izquierda
= *=	Asignación	De derecha a izquierda

© 2000-2025, MetaQuotes Ltd.

Operación	Descripción	Orden de ejecución
/= %= += -= <<= >>= &= ^= |=	Multiplicación con asignación División con asignación División inexacta con asignación Suma con asignación Resta con asignación Desplazamiento a la izquierda con asignación Desplazamiento a la derecha con asignación A nivel de bits AND con asignación Excluyendo OR con asignación A nivel de bits OR con asignación	
,	Coma	De izquierda a derecha

Para cambiar el orden de ejecución de la operación se utilizan los paréntesis que tengan la más alta prioridad.

© 2000-2025, MetaQuotes Ltd.

Operadores

Los operadores del lenguaje describen algunas acciones algorítmicas que tienen que ser ejecutadas para resolver el problema. El cuerpo del programa es una secuencia de estos operadores. Los operadores siguen uno detrás del otro y se separan con el punto y coma.

Operador	Descripción
Operador compuesto {}	Uno o más operadores de cualquier tipo encerrados entre las llaves { }
Operador-expresión (;)	Cualquier expresión que se termina con el punto y coma (;)
Operador return	Termina la ejecución de la función corriente y devuelve el control al programa iniciador
Operador condicional if-else	Se usa si hay necesidad de hacer una elección
Operador condicional ?:	Un análogo más simple del operador condicional if-else
Operador de selección switch	Pasa el control al operador que corresponde al valor de la expresión
Operador cíclico while	Ejecuta un operador hasta que la expresión verificada no sea falsa. El chequeo de la expresión se realiza antes de cada iteración
Operador cíclico for	Ejecuta un operador hasta que la expresión verificada no sea falsa. El chequeo de la expresión se realiza antes de cada iteración
Operador cíclico do-while	Ejecuta un operador hasta que la expresión verificada no sea falsa. El chequeo de la condición del fin se realiza al final, después de cada ronda del ciclo. El cuerpo del ciclo se ejecuta por lo menos una vez.
Operador break	Suspende la ejecución del operador externo adjunto más próximo switch, while, do-while o for
Operador continue	Pasa el control al inicio del operador cíclico externo más próximo while, do-while o for
Operador @	Implementa la multiplicación de matrices según las reglas del álgebra lineal. Permite realizar el producto de matrices, vectores y el producto escalar de vectores.
Operador new	Crea el objeto del tamaño correspondiente y devuelve el descriptor del objeto creado.
Operador delete	Elimina el objeto creado por el operador new.

Un operador puede ocupar una o más líneas. Dos o más operadores pueden ubicarse en una línea. Los operadores que controlan el orden de ejecución (if, if-else, switch, while y for) pueden ser insertados uno en otro.

Ejemplo:

```
if(Month() == 12)
```

© 2000-2025, MetaQuotes Ltd.

```
if(Day() == 31) Print("Happy New Year!");
```

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador compuesto

Operador compuesto (bloque) se compone de uno o más operadores de cualquier tipo encerrados entre las llaves { }. Después de la llave que cierra no puede haber el punto y coma (;).

Ejemplo:

```
if(x==0)
  {
   Print("invalid position x = ",x);
   return;
  }
```

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador-expresión

Cualquier expresión que se termina con punto y coma (;) es un operador. Más abajo vienen los ejemplos de los operadores-expresiones.

Operador de asignación:

Identificador = expresión;

```
x=3;
y=x=3;
bool equal=(x==y);
```

En una expresión el operador de asignación puede usarse varias veces. En este caso, el procesamiento de la expresión se realiza de derecha a izquierda.

Operador de llamada a función:

Nombre_de_función (argumento1,..., argumentoN);

```
FileClose(file);
```

Operador vacío

Se compone únicamente del punto y coma (;) y se usa para determinar el cuerpo vacío del operador de control.

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador de retorno return

El operador return termina la ejecución de la función corriente y devuelve el control al programa iniciador. El resultado del cálculo de expresión es retornado a la función invocada. La expresión puede contener un operador de asignación.

Ejemplo:

```
int CalcSum(int x, int y)
  {
   return(x+y);
  }
```

En las funciones con tipo de valor devuelto void es necesario usar el operador return sin expresión:

```
void SomeFunction()
  {
   Print("Hello!");
   return;     // se puede eliminar este operador
  }
```

La llave derecha de la función comprende la ejecución implícita del operador return sin expresión.

Se puede devolver los tipos simples, estructuras simples, punteros a objetos. Mediante el operador *return* no se puede devolver cualquier array, objetos de clases, variables del tipo de estructuras compuestas.

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador condicional if-else

El operador IF - ELSE se usa cuando surge la necesidad de hacer una elección. Formalmente, la sintaxis es así:

```
if (expresión)
     operador1
else
     operador2
```

Si la expresión es verdadera, se ejecuta el operador1 y el control se pasa al operador que sigue después del operador2 (es decir, el operador2 no se ejecuta). Si la expresión es falsa, se ejecuta el operador2.

La parte else del operador if se puede omitir. Por eso puede surgir una ambigüedad en los operadores interiores if con la parte omitida else. En este caso, else se vincula con anterior operador if más cercano en el mismo bloque, y que no tiene la parte else.

Ejemplos:

```
//--- La parte else se refiere al segundo operador if:
if(x>1)
   if(y==2) z=5;
else     z=6;
//--- La parte else se refiere al primer operador if
if(x>1)
  {
   if(y==2) z=5;
  }
else        z=6;
//--- Operador interior
if(x=='a')
  {
   y=1;
  }
else if(x=='b')
  {
   y=2;
   z=3;
  }
else if(x=='c')
  {
   y=4;
  }
else Print("ERROR");
```

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador condicional ?:

La forma general de un operador ternario es la siguiente:

```
expresión1? expresión2:expresión3
```

Como el primer operando - "expresión1" se puede utilizar cualquier expresión, el resultado de la cual es el valor del tipo bool. Si el resultado es true, entonces se ejecuta el operador asignado por el segundo operando, es decir, "expresión2".

Si el primer operando es igual a false, entonces se ejecuta el tercer operando - "expresión3". El segundo y el tercer operando, es decir, "expresión2" y "expresión3" tienen que retornar los valores del mismo tipo y no deben tener el tipo void. El resultado de ejecución del operador condicional es el resultado de la expresión2 o el de la expresión3, dependiendo del resultado de la expresión1.

```
//--- normalizamos la diferencia entre los precios de apertura y de cierre para el ran
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

Esta entrada es equivalente a la siguiente

```
   double true_range;
   if(High==Low)true_range=0;                // si High y Low son iguales
   else true_range=(Close-Open)/(High-Low); // si el rango no es cero
```

Limitaciones en el uso del operador

A base del valor "expresión1" el operador tiene que devolver uno de dos valores - "expresión2" o "expresión3". Existe una serie de limitaciones para estas expresiones:

1. No se puede confundir el tipo definido por el usuario con el tipo simple o enumeración. Para el puntero se puede utilizar NULL.
2. Si los tipos de los valores son simples, entonces el tipo del operador será el tipo máximo (véase Conversión de tipos).
3. Si uno de los valores tiene el tipo de enumeración y el segundo es del tipo numérico, la enumeración se reemplace con int y se aplica la segunda regla.
4. Si los dos valores son valores de enumeración, entonces sus tipos tienen que ser iguales, y el tipo del operador será la enumeración.

Limitaciones para los tipos definidos por el usuario (clases o estructuras):

a) los tipos tienen que ser idénticos o uno debe heredarse del otro.

b) si los tipos no son idénticos (la herencia), entonces el hijo se convierte al padre de forma implícita, es decir el tipo del operador va a ser del tipo del padre.

c) no se puede confundir el objeto y el puntero - o las dos expresiones son objetos, o bien los punteros. Se puede usar NULL para el puntero.

Nota

Tenga cuidado a la hora de usar el operador condicional como un argumento de la función sobrecargada, porque el tipo del resultado del operador condicional se determina en el momento de

© 2000-2025, MetaQuotes Ltd.

compilación del programa. Y este tipo se determina como el más grande de los tipos "expresión2" y "expresión3".

Ejemplo:

```
void func(double d) { Print("double argument: ",d); }
void func(string s) { Print("string argument: ",s); }

bool   Expression1=true;
double Expression2=M_PI;
string Expression3="3.1415926";

void OnStart()
  {
   func(Expression2);
   func(Expression3);

   func(Expression1?Expression2:Expression3);    // recibimos un aviso del compilador s
   func(!Expression1?Expression2:Expression3);   // recibimos un aviso del compilador s
  }

//   Resultado:
//   double argument: 3.141592653589793
//   string argument: 3.1415926
//   string argument: 3.141592653589793
//   string argument: 3.1415926
```

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador switch

Compara el valor de expresión con las constantes en todas las variantes de *case* y pasa el control al operador correspondiente al valor de la expresión. Cada variante *case* puede ser marcada con la constante entera, constante de signo o expresión constante. La expresión constante no puede contener las variables o llamadas a funciones. La expresión del operador *switch* tiene que ser del tipo entero - int o uint

```
switch(expresión)
  {
   case constante: operadores
   case constante: operadores
      ...
   default: operadores
  }
```

Los operadores vinculados a la marca *default* se ejecutan si ninguna de las constantes en los operadores *case* no es igual al valor de la expresión. La variante *default* no tiene que ser declarada obligatoriamente y no tiene que ser última obligatoriamente. Si ninguna de las constantes corresponde al valor de la expresión y la variante *default* no está presente, no se ejecuta ninguna acción.

La palabra clave *case* y la constante son sólo marcas, y si los operadores se ejecutan para una variante *case*, entonces más adelante van a ejecutarse los operadores de todas las variantes posteriores hasta que se llegue al operador *break*. Eso permite vincular una sucesión de operadores con varias variantes.

Una expresión constante se calcula durante el proceso de compilación. Nunca dos constantes en el mismo operador *switch* pueden tener los mismos valores.

Ejemplos:

```
//--- el primer ejemplo
switch(x)
  {
   case 'A':
      Print("CASE A");
      break;
   case 'B':
   case 'C':
      Print("CASE B or C");
      break;
   default:
      Print("NOT A, B or C");
      break;
  }

//---  el segundo ejemplo
   string res="";
   int i=0;
   switch(i)
     {
      case 1:
```

© 2000-2025, MetaQuotes Ltd.

```
         res=i;break;
      default:
         res="default";break;
      case 2:
         res=i;break;
      case 3:
         res=i;break;
     }
   Print(res);
/*
   Resultado
   default
*/
```

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador cíclico while

El operador while se compone de la expresión chequeada y el operador que tiene que ser ejecutado:

```
while(expresión)
  operador;
```

Si la expresión es verdadera, entonces el operador se ejecuta hasta que la expresión se haga falsa. Si la expresión es falsa, entonces el control se pasa al siguiente operador. El valor de la expresión se define antes de que el operador sea ejecutado. Por tanto, si la expresión es falsa desde el principio, entonces el operador no será ejecutado en absoluto.

Nota

Si se supone procesar una gran cantidad de iteraciones en el ciclo, entonces se recomienda comprobar el hecho de finalización forzosa del programa por medio de la función IsStopped().

Ejemplo:

```
while(k<n  && !IsStopped())
  {
   y=y*x;
   k++;
  }
```

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador cíclico for

El operador for se compone de tres expresiones y un operador ejecutable:

```
for(expresión1; expresión2; expresión3)
   operador;
```

Expresión1 describe la inicialización del ciclo. Expresión2 comprueba la condición del fin del ciclo. Si es verdadera, el operador del cuerpo de ciclo for se ejecuta. Todo se repite hasta que la expresión2 se haga falsa. Si es falsa, el ciclo se termina y el control se pasa al siguiente operador. Expresión3 se calcula después de cada iteración.

El operador for equivale a la siguiente sucesión de operadores:

```
expresión1;
while(expresión2)
  {
   operador;
   expresión3;
  };
```

En operador for puede faltar cualquiera de las tres expresiones o todas, sin embargo, no se puede omitir los puntos y comas (;) que las separan. Si la expresión2 se omite, se considera que es siempre verdadera. El operador for(;;) es un ciclo infinito equivalente al operador while(1). Cada expresión1 y 3 puede componerse de varias expresiones unidas por el operador coma ','.

Nota

Si se supone procesar una gran cantidad de iteraciones en el ciclo, entonces se recomienda comprobar el hecho de finalización forzosa del programa por medio de la función IsStopped().

Ejemplos:

```
for(x=1;x<=7000; x++)
  {
   if(IsStopped())
      break;
   Print(MathPower(x,2));
  }
//--- otro ejemplo
for(;!IsStopped();)
  {
   Print(MathPower(x,2));
   x++;
   if(x>10) break;
  }
//--- el tercer ejemplo
for(i=0,j=n-1;i<n && !IsStopped();i++,j--) a[i]=a[j];
```

Véase también

© 2000-2025, MetaQuotes Ltd.

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador cíclico do while

Los ciclos for y while realizan la comprobación de terminación al principio del ciclo y no en su final. El tercer operador del ciclo do - while comprueba la condición de terminación al final, después de cada iteración del ciclo. El cuerpo del ciclo siempre se ejecuta por lo menos una vez.

```
do
   operador;
while(expresión);
```

Al principio se ejecuta el operador, luego se calcula la expresión. Si la expresión es verdadera, entonces el operador vuelve a ejecutarse, etc. Si la expresión se hace falsa, el ciclo se acaba.

Nota

Si se supone procesar una gran cantidad de iteraciones en el ciclo, entonces se recomienda comprobar el hecho de finalización forzosa del programa por medio de la función IsStopped().

Ejemplo:

```
//--- el cálculo de la sucesión de Fibonacci
   int counterFibonacci=15;
   int i=0,first=0,second=1;
   int currentFibonacciNumber;
   do
     {
      currentFibonacciNumber=first+second;
      Print("i = ",i,"  currentFibonacciNumber = ",currentFibonacciNumber);
      first=second;
      second=currentFibonacciNumber;
      i++; // Sin este operador se obtiene el ciclo infinito!
     }
   while(i<counterFibonacci && !IsStopped());
```

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador break

El operador break termina la ejecución del más cercano operador adjunto externo switch, while, do-while o for. El control se pasa al operador que sigue después del terminado. Uno de los objetivos de este operador consiste en terminar el ciclo al adjudicar un valor concreto a una variable.

Ejemplo:

```
//--- la búsqueda del primer elemento cero
for(i=0;i<array_size;i++)
  if(array[i]==0)
    break;
```

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador de continuación continue

El operador continue pasa el control al inicio del operador externo de ciclo más cercano while, do-while o for, provocando así el comienzo de la siguiente iteración. Por su acción, este operador es contrario al operador break.

Ejemplo:

```
//--- suma de todos los elementos no nulos
int func(int array[])
  {
   int array_size=ArraySize(array);
   int sum=0;
   for(int i=0;i<array_size; i++)
     {
      if(a[i]==0) continue;
      sum+=a[i];
     }
   return(sum);
  }
```

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador de producto matricial @

El operador @ en lenguaje MQL5 implementa la multiplicación de matrices según las reglas del álgebra lineal. Permite realizar el producto de matrices, vectores y el producto escalar de vectores.

Tipos soportados de elementos:.

- float
- double
- complex<float>
- complex<double>

Importante: los tipos de elemento de los operandos izquierdo y derecho deben coincidir.

Ejemplos de uso

1. Multiplicación de matrices (matriz × matriz)

```
matrix A(2, 3);
matrix B(3, 2);
matrix C = A @ B; // Resultado: matriz C con un tamaño [2,2]
```

2. Multiplicación matricial (matriz × vector)

```
matrix M(2, 3);
vector V(3);
vector R = M @ V; // Resultado: vector R de 2 elementos
```

3. Multiplicación de matrices (vector × matriz)

```
matrix M(2, 3);
vector V(1, 2);
vector R = V @ M; // Resultado: vector R de 3 elementos
```

4. Producto escalar (vector × vector)

```
vector V1(1, 3), V2(1, 3);
double r = V1 @ V2; // Resultado: escalar
```

Observación

Las dimensiones deben seguir las reglas de multiplicación: número de columnas del primer operando = número de filas del segundo operando.

Si las dimensiones son incorrectas, se mostrará una excepción en tiempo de ejecución.

Los vectores de la izquierda se consideran horizontales (1×n).

© 2000-2025, MetaQuotes Ltd.

Los vectores de la derecha se consideran verticales (n×1).

La prioridad de la definición @ corresponde a la prioridad de multiplicación, es decir, para una entrada de la forma D=C+A@B, se realizará primero la multiplicación matricial T=A@B y después la suma elemento a elemento D=C+T.

Ver también

MatMul, GeMM, Kron, Dot,

© 2000-2025, MetaQuotes Ltd.

Operador-creador de objetos new

El operador new crea automáticamente un objeto de tamaño correspondiente, arranca el constructor del objeto y devuelve el descriptor del objeto creado. En caso de un fallo el operador devuelve el descriptor cero que se puede comparar con la constante NULL.

El operador *new* puede ser aplicado únicamente a los objetos de la clase, no se puede aplicarlo a las estructuras.

El operador no se aplica para crear los arrays de objetos. Para eso hay que usar la función ArrayResize().

Ejemplo:

```
//+------------------------------------------------------------------+
//| Creación de una figura                                           |
//+------------------------------------------------------------------+
void CTetrisField::NewShape()
  {
   m_ypos=HORZ_BORDER;
//--- de una manera aleatoria creamos una de 7 posibles figuras
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- dibujamos
   if(m_shape!=NULL)
     {
      //--- configuraciones iniciales
      m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
      m_shape.SetYPos(m_ypos);
      m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
      //--- dibujamos
      m_shape.Draw();
     }
//---
  }
```

Cabe mencionar que el descriptor del objeto no es un puntero a la memoria.

El objeto creado con operador new tiene que ser eliminado de una forma explícita por el operador delete.

Véase también

© 2000-2025, MetaQuotes Ltd.

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Operador-eliminador de objetos delete

El operador delete elimina un objeto creado por el operador new, arranca el destructor de clase correspondiente y libera la memoria ocupada por el objeto. El descriptor real de un objeto existente se utiliza como un operando. Una vez terminada la operación delete el descriptor del objeto pierde su validez.

Ejemplo:

```
      //--- eliminamos la figura colocada
      delete m_shape;
      m_shape=NULL;
      //--- creamos una figura nueva
      NewShape();
```

Véase también

Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Funciones

Cualquier tarea puede ser dividida en unas subtareas, cada una de las cuales bien puede ser representada directamente en forma de un código o bien dividida en subtareas más pequeñas. Este método se llama *refinamiento sucesivo*. Las funciones sirven para escribir el código de programación de estas tareas a resolver. El código que describe lo que hace la función se llama *definición de función:*

```
encabezado_de_función
  {
   instrucción
  }
```

Todo lo que se encuentra antes de la primera llave es el encabezado de la definición de función, lo que viene entre las llaves es *el cuerpo* de la definición de función. El encabezado de la función contiene la descripción del tipo de valor devuelto, nombre del (identificador) y parámetros formales. La cantidad de parámetros traspasados a una función está limitada y no puede superar 64.

Se puede llamar a la función de otras partes del programa tantas veces como haga falta. En realidad, el tipo devuelto, identificador de la función y tipos de parámetros constituyen *prototipo de función*.

Un prototipo de función es la declaración de una función pero no su definición. Gracias a la declaración explícita del tipo devuelto y de la lista de tipos de argumentos, la prueba rigurosa de tipos y las conversiones implícitas de tipos son posibles durante la invocación de las funciones. Sobre todo, las declaraciones de funciones se utilizan muy a menudo en las clases para mejorar la lectura del código fuente.

La definición de función tiene que corresponder exactamente a su declaración. Cada función declarada tiene que estar definida.

Ejemplo:

```
double                          // tipo del valor devuelto
linfunc (double a, double b) // nombre de función y lista de parámetros
  {
                                // operador compuesto
   return (a + b);              // valor devuelto
  }
```

El operador return puede devolver el valor de la expresión que se encuentra en este operador. Si hace falta, el valor de la expresión se convierte al tipo del resultado de función. Se puede devolver los tipos simples, estructuras simples, punteros a objetos. Mediante el operador *return* no se puede devolver cualquier array, objetos de clases, variables del tipo de estructuras compuestas.

La función que no devuelve el valor tiene que ser descrita como la que tiene el tipo void.

Ejemplo:

```
void errmesg(string s)
  {
   Print("error: "+s);
  }
```

© 2000-2025, MetaQuotes Ltd.

Los parámetros traspasados a la función pueden tener los valores por defecto que se definen por las constantes del tipo correspondiente.

Ejemplo:

```
int somefunc(double a,
             double d=0.0001,
             int n=5,
             bool b=true,
             string s="passed string")
  {
   Print("Parámetro obligatorio a=",a);
   Print("Se han pasado los siguientes parámetros: d = ",d," n = ",n," b = ",b," s = "
   return(0);
  }
```

Si algún parámetro ha obtenido el valor default, todos los parámetros siguientes también deben tener el valor default.

Ejemplo de una declaración errónea:

```
int somefunc(double a,
             double d=0.0001,    // se ha declarado el valor por defecto 0.0001
             int n,              // el valor por defecto no aparece !
             bool b,             // el valor por defecto no aparece !
             string s="passed string")
  {
  }
```

Véase también

Sobrecarga, Funciones virtuales, Polimorfismo

© 2000-2025, MetaQuotes Ltd.

Llamada a una función

Si un nombre que no ha sido descrito antes aparece en la expresión y le sigue el paréntesis izquierdo, entonces según el contexto éste se considera el nombre de una función.

```
nombre_de_función (x1, x2,..., xn)
```

Los argumentos (parámetros formales) se traspasan por valor, es decir, cada expresión x1, . . . , xn se calcula, y su valor es pasado a la función. El orden del cálculo de expresión y el orden de la carga de valores no se garantizan. Durante la ejecución se realiza el chequeo del número y el tipo de argumentos pasados a la función. Este modo de dirigirse a una función se llama la llamada por valor.

La llamada a una función es una expresión cuyo valor es el valor devuelto por la función. El tipo de función descrito tiene que corresponder al tipo del valor devuelto. La función puede ser declarada o descrita en cualquier parte del programa a nivel global, es decir, fuera de otras funciones. Una función no puede ser declarada o descrita dentro de otra función.

Ejemplos:

```
int start()
  {
   double some_array[4]={0.3, 1.4, 2.5, 3.6};
   double a=linfunc(some_array, 10.5, 8);
   //...
  }
double linfunc(double x[], double a, double b)
  {
   return (a*x[0] + b);
  }
```

Al invocar una función que tiene los parámetros predefinidos, la lista de parámetros pasados puede ser limitada, pero no antes del primer parámetro predefinido.

Ejemplos:

```
void somefunc(double init,
              double sec=0.0001, //los valores por defecto están definidos
              int level=10);
//...
somefunc();                      // llamada errónea. tiene que haber el primer paráme
somefunc(3.14);                  // llamada correcta
somefunc(3.14,0.0002);           // llamada correcta
somefunc(3.14,0.0002,10);        // llamada correcta
```

Al llamar a una función, no se puede omitir los parámetros, aunque éstos tengan los valores predefinidos:

```
somefunc(3.14, , 10);           // llamada errónea -> el segundo parámetro está omiti
```

Véase también

Sobrecarga, Funciones virtuales, Polimorfismo

© 2000-2025, MetaQuotes Ltd.

Traspaso de parámetros

Existen dos métodos mediante los cuales el lenguaje de programación puede pasar el argumento al subprograma (función). El primer modo consiste en traspasar los parámetros por valor. Este método copia el valor del argumento en el parámetro formal de la función. Por tanto, cualquieras que sean las modificaciones de este parámetro dentro de la función, no tendrán ninguna influencia sobre el correspondiente argumento de llamada.

```
//+------------------------------------------------------------------+
//| traspaso de parámetros por valor                                 |
//+------------------------------------------------------------------+
double FirstMethod(int i,int j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a y b antes de la llamada:",a," ",b);
   double d=FirstMethod(a,b);
   Print("a y b después de la llamada:",a," ",b);
  }
//--- resultado de ejecución del script
//  a y b antes de la llamada: 14 8
//  a y b después de la llamada: 14 8
```

El segundo modo es el traspaso de parámetros por referencia. En este caso la referencia al parámetro (y no su valor) se pasa al parámetro de la función. Se usa dentro de la función para dirigirse al parámetro actual indicado en la llamada. Esto significa que las modificaciones del parámetro van a influir sobre el argumento utilizado para invocar la función.

```
//+------------------------------------------------------------------+
//| traspaso de parámetros por referencia                            |
//+------------------------------------------------------------------+
double SecondMethod(int &i,int &j)
  {
   double res;
//---
   i*=2;
   j/=2;
```

© 2000-2025, MetaQuotes Ltd.

```
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a y b antes de la llamada:",a," ",b);
   double d=SecondMethod(a,b);
   Print("a y b después de la llamada:",a," ",b);
  }
//+------------------------------------------------------------------+
//--- resultado de ejecución del script
//  a y b antes de la llamada: 14 8
//  a y b después de la llamada: 28 4
```

MQL5 utiliza los dos métodos, salvo una excepción: los arrays, variables del tipo de estructuras y objetos de clases siempre se traspasan por referencia. Para excluir los cambios de los parámetros actuales (argumentos pasados al invocar una función) es necesario usar el especificador de acceso const. Cuando se intenta cambiar el contenido de la variable que ha sido declarada con el especificador *const*, el compilador mostrará un error.

Nota

Hay que recordar que los parámetros se traspasan a una función del revés, es decir, al principio se calcula y se pasa el último parámetro, luego el penúltimo, etc. En último lugar se calcula y se pasa el parámetro que está primero después de las llaves.

Ejemplo:

```
void OnStart()
  {
//---
   int a[]={0,1,2};
   int i=0;

   func(a[i],a[i++],"La primera llamada(i = "+string(i)+")");
   func(a[i++],a[i],"La segunda llamada(i = "+string(i)+")");
// Resultado:
// Primera llamada(i=0) : par1 = 1     par2 = 0
// Segunda llamada(i=1) : par1 = 1     par2 = 1

  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void func(int par1,int par2,string comment)
  {
   Print(comment,": par1 = ",par1,"    par2 = ",par2);
  }
```

Durante la primera llamada en el ejemplo mencionado, al principio la variable ***i*** toma parte en la concatenación de cadenas de caracteres

```
   "La primera llamada(i = "+string(i)+")"
```

además allí su valor no se cambia. Luego la variable **i** toma parte en el calculo del elemento del array ***a[i++]***, es decir, después de tomar el elemento i del array, la variable ***i*** se incrementa. Y sólo después de eso se calcula el primer parámetro con el valor cambiado de la variable ***i***.

En la segunda llamada durante el cálculo de todos los tres parámetros, se usa el mismo valor de i que ha sido cambiado durante la primera invocación de la función, y sólo una vez calculado el primer parámetro, la variable ***i*** vuelve a cambiarse.

Véase también

Visibilidad y tiempo de vida de variables, Sobrecarga, Funciones virtuales, Polimorfismo

© 2000-2025, MetaQuotes Ltd.

Sobrecarga de funciones

Por lo común, en el nombre de una función intentan reflejar su principal finalidad. Por regla general, los programas legibles contienen diversos identificadores bien seleccionados. A veces, diferentes funciones se usan con los mismos propósitos. Vamos a considerar, por ejemplo, la función que calcula el valor promedio del array de números de doble precisión, y la misma función pero la que opera con el array de números enteros. Es cómodo nombrar ambas funciones AverageFromArray:

```
//+------------------------------------------------------------------+
//| cálculo del promedio para el array de tipo double                |
//+------------------------------------------------------------------+
double AverageFromArray(const double & array[],int size)
  {
   if(size<=0) return 0.0;
   double sum=0.0;
   double aver;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];    // suma para double
     }
   aver=sum/size;       // simplemente dividimos la suma por la cantidad
//---
   Print("Cálculo de la media para un array del tipo double");
   return aver;
  }
//+------------------------------------------------------------------+
//| cálculo del promedio para el array de tipo int                   |
//+------------------------------------------------------------------+
double AverageFromArray(const int & array[],int size)
  {
   if(size<=0) return 0.0;
   double aver=0.0;
   int sum=0;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];     // suma para int
     }
   aver=(double)sum/size;// convertimos la suma al tipo double y la dividimos
//---
   Print("Cálculo de la media para un array del tipo int");
   return aver;
  }
```

Cada función contiene la extracción del mensaje a través de la función Print();

```
   Print("Cálculo de la media para un array del tipo int");
```

El compilador elige una función necesaria de acuerdo con los tipos de argumentos y su cantidad. La regla, según la cual se hace la elección, se llama *algoritmo de equivalencia a la signatura*. Una signatura es una lista de tipos usados en la declaración de función.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int    a[5]={1,2,3,4,5};
   double b[5]={1.1,2.2,3.3,4.4,5.5};
   double int_aver=AverageFromArray(a,5);
   double double_aver=AverageFromArray(b,5);
   Print("int_aver = ",int_aver,"   double_aver = ",double_aver);
  }
//--- Resultado de funcionamiento del script
// cálculo del promedio para el array de tipo int
// cálculo del promedio para el array de tipo double
// int_aver = 3.00000000    double_aver = 3.30000000
```

La sobrecarga es la práctica de asignación de varios valores a una función. La elección de un valor específico depende de los tipos de argumentos recibidos por la función. Una función específica se elige a base de la correspondencia de la lista de argumentos durante la invocación de la función a la lista de parámetros en la declaración de la función.

Cuando se invoca una función sobrecargada, el compilador debe tener un algoritmo para elegir una función apropiada. El algoritmo que hace la elección depende del tipo de conversiones que estén presentes. La mejor correlación tiene que ser única. Tiene que ser la mejor por lo menos para uno de los argumentos, e igual de buena como las demás correspondencias para todos los otros argumentos.

Más abajo viene el algoritmo de correspondencia para cada argumento.

Algoritmo de elección de una función sobrecargada

1. Usar la correspondencia estricta (si es posible).
2. Intentar el incremento estándar del tipo.
3. Intentar la transformación estándar del tipo.

El incremento estándar del tipo es mejor que las demás transformaciones estándar. El incremento es una transformación float en double, y también bool, char, short o enum en int. Además, a las transformaciones estándar pertenecen las transformaciones de los arrays de tipos enterossimilares. Los tipos similares son los siguientes: bool, char, uchar, puesto que estos tres tipos son enteros de un byte; los enteros de dos bytes short y ushort; los enteros de 4 bytes int, uint y color; long, ulong y datetime.

No hay duda que la correspondencia estricta es la mejor. Para conseguirla se puede usar las coversiones. El compilador no podrá con una situación de doble sentido, así que no hay que fiarse de las ligeras diferencias en los tipos y las transformaciones implícitas que hacen la función sobrecargada poco clara.

Si Usted tiene duda, recurra a las conversiones explícitas para conseguir la correspondencia estricta.

Como ejemplo de las funciones sobrecargadas en MQL5 puede servir el de la función ArrayInitialize().

Las reglas de sobrecarga de funciones también se aplican a la sobrecarga de métodos de clases.

© 2000-2025, MetaQuotes Ltd.

La sobrecarga de las funciones del sistema está permitida pero en este caso hay que mirar que el compilador pueda elegir sin problema alguno la función necesaria. Como ejemplo, podemos sobrecargar la función del sistema fmax() de 3 formas distintas, pero sólo dos opciones van a ser correctas.

Ejemplo:

```
// sobrecarga está permitida - se diferencia por la cantidad de parámetros
double fmax(double a,double b,double c);

// sobrecarga con error
// la cantidad de parámetros es diferente, pero el último tiene el valor por defecto
// eso lleva a la ocultación de la función del sistema durante la llamada, pero esto e
double fmax(double a,double b,double c=DBL_MIN);

// sobrecarga normal por el tipo de parámetro
int fmax(int a,int b);
```

Véase también

Sobrecarga, Funciones virtuales, Polimorfismo

© 2000-2025, MetaQuotes Ltd.

Sobrecarga de operaciones

Para que la lectura y escritura del código sea más cómoda, se permite la sobrecarga de algunas operaciones. El operador de sobrecarga se escribe con la palabra clave operator. Está permitida la sobrecarga de las siguientes operaciones:

- binarias +,-,/,*,%,<<,>>,==,!=,<,>,<=,>=,=,+=,-=,/=,*=,%=,&=,|=,^=,<<=,>>=,&&,||,&,|,^;
- unarias +,-,++,--,!,~;
- operador de asignación =;
- operador de indexación [].

La sobrecarga de operaciones permite usar la notación operacional (anotación en forma de expresiones simples) con objetos complejos: estructuras y clases. La escritura de expresiones con el uso de las operaciones sobrecargadas facilita la percepción del código fuente, puesto que la implementación más compleja está oculta.

Como ejemplo vamos a considerar los números complejos, de amplio uso en las matemáticas, que se componen de la parte real e imaginaria. En el lenguaje MQL5 no hay un tipo de datos para representar los números complejos pero hay una posibilidad de crear un nuevo tipo de datos en forma de una estructura o clase. Vamos a declarar una estructura complex y definir dentro de ella cuatro métodos que realizan cuatro operaciones aritméticas:

```
//+------------------------------------------------------------------+
//| Estructura para operaciones con números complejos                |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // parte real
   double            im; // parte imaginaria
   //--- constructores
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- operaciones aritméticas
   complex           Add(const complex &l,const complex &r) const;  // suma
   complex           Sub(const complex &l,const complex &r) const;  // resta
   complex           Mul(const complex &l,const complex &r) const;  // multiplicación
   complex           Div(const complex &l,const complex &r) const;  // división
  };
```

Ahora podemos declarar en nuestro código las variables que representan los números complejos, y trabajar con ellas.

Por ejemplo:

```
void OnStart()
  {
//--- declaramos e inicializamos las variables del tipo complejo
   complex a(2,4),b(-4,-2);
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- sumamos dos números
   complex z;
   z=a.Add(a,b);
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- multiplicamos dos números
   z=a.Mul(a,b);
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- dividimos dos números
   z=a.Div(a,b);
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
```

Pero para las operaciones aritméticas habituales con números complejos sería más cómodo utilizar los operadores habituales "+","-","*" y "/".

La palabra clave operator se utiliza para definir la función miembro que realiza la conversión del tipo. Las operaciones unarias y binarias para las variables-objetos de la clase pueden ser sobrecargadas como las funciones miembros no estáticas. Actúan de forma implícita sobre el objeto de la clase.

La mayoría de las operaciones binarias pueden ser sobrecargadas como funciones ordinarias que aceptan uno o ambos argumentos en forma de la variable de la clase o en forma del puntero al objeto de esta clase. Para nuestro tipo complex la sobrecarga en la declaración va a ser la siguiente:

```
   //--- operadores
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
```

Ejemplo completo del script:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos e inicializamos las variables del tipo complejo
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
   //a.re=5;
   //a.im=1;
   //b.re=-1;
   //b.im=-5;
//--- sumamos dos números
   complex z=a+b;
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- multiplicamos dos números

   z=a*b;
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- dividimos dos números
   z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
//+------------------------------------------------------------------+
//| Estructura para las operaciones con números complejos            |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // parte real
   double            im; // parte imaginaria
   //--- constructores
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- operaciones aritméticas
   complex           Add(const complex &l,const complex &r) const;  // suma
   complex           Sub(const complex &l,const complex &r) const;  // resta
   complex           Mul(const complex &l,const complex &r) const;  // multiplicación
   complex           Div(const complex &l,const complex &r) const;  // división
   //--- operadores binarios
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
  };
//+------------------------------------------------------------------+
//|  Suma                                                            |
//+------------------------------------------------------------------+
complex complex::Add(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re+r.re;
   res.im=l.im+r.im;
//--- resultado
   return res;
  }
//+------------------------------------------------------------------+
//|   Resta                                                          |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
complex complex::Sub(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re-r.re;
   res.im=l.im-r.im;
//--- resultado
   return res;
  }
//+------------------------------------------------------------------+
//|   Multiplicación                                                 |
//+------------------------------------------------------------------+
complex complex::Mul(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re*r.re-l.im*r.im;
   res.im=l.re*r.im+l.im*r.re;
//--- resultado
   return res;
  }
//+------------------------------------------------------------------+
//|    División                                                      |
//+------------------------------------------------------------------+
complex complex::Div(const complex &l,const complex &r) const
  {
//--- número complejo vacío
   complex res(EMPTY_VALUE,EMPTY_VALUE);
//--- comprobación del cero
   if(r.re==0 && r.im==0)
     {
      Print(__FUNCTION__+": number is zero");
      return(res);
     }
//--- variables auxiliares
   double e;
   double f;
//--- selección del variante de cálculo
   if(MathAbs(r.im)<MathAbs(r.re))
     {
      e = r.im/r.re;
      f = r.re+r.im*e;
      res.re=(l.re+l.im*e)/f;
      res.im=(l.im-l.re*e)/f;
     }
   else
     {
      e = r.re/r.im;
      f = r.im+r.re*e;
      res.re=(l.im+l.re*e)/f;
      res.im=(-l.re+l.im*e)/f;
     }
//--- resultado
   return res;
  }
```

© 2000-2025, MetaQuotes Ltd.

La mayoría de las operaciones unarias para las clases pueden ser sobrecargadas como funciones ordinarias que aceptan el único argumento objeto de la clase o puntero a él. Vamos a agregar la sobrecarga de operaciones unarias "-" y "!".

```
//+------------------------------------------------------------------+
//| Estructura para las operaciones con números complejos            |
//+------------------------------------------------------------------+
struct complex
  {
   double            re;       // parte real
   double            im;       // parte imaginaria
...
   //--- operadores unarios
   complex operator-()  const; // menos unario
   bool    operator!()  const; // negación
  };
...
//+------------------------------------------------------------------+
//|   Sobrecarga del operador "menos unario"                         |
//+------------------------------------------------------------------+
complex complex::operator-() const
  {
   complex res;
//---
   res.re=-re;
   res.im=-im;
//--- resultado
   return res;
  }
//+------------------------------------------------------------------+
//|    Sobrecarga del operador "negación lógica"                     |
//+------------------------------------------------------------------+
bool complex::operator!() const
  {
//--- ¿es igual a cero la parte real e imaginaria del número complejo?
   return (re!=0 && im!=0);
  }
```

Ahora podemos comprobar el valor del número complejo respecto al cero y obtener valor negativo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos e inicializamos las variables del tipo complejo
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- dividimos dos números
   complex z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//--- por defecto, el número complejo es igual a cero (en el constructor por defecto
   complex zero;
   Print("!zero=",!zero);
//--- asignamos valor negativo
   zero=-z;
   PrintFormat("z=%.2f+i*%.2f,  zero=%.2f+i*%.2f",z.re,z.im, zero.re,zero.im);
   PrintFormat("-zero=%.2f+i*%.2f",-zero.re,-zero.im);
//--- volvemos a comprobar la igualdad a cero
   Print("!zero=",!zero);
//---
  }
```

Fíjense que en este caso no hemos tenido la necesidad de sobrecargar la operación de asignación "=", porque las estructuras de tipos simples se puede copiar una a otra directamente. De esta manera, ahora podemos escribir el código para los cálculos que incluyen los números complejos en una manera a la que estamos acostumbrados.

La sobrecarga del operador de indexación permite obtener los valores de los arrays encerrados en un objeto de una manera más sencilla y habitual, y eso también contribuye a la mejor legibilidad y comprensión del código fuente de los programas. Por ejemplo, tenemos que asegurar el acceso a un símbolo en la cadena, según la posición especificada. Una cadena en el lenguaje MQL5 es un tipo separado string, que no es un array de símbolos. Pero mediante la operación de indexación sobrecargada podemos asegurar un trabajo sencillo y transparente en la clase creada CString:

```
//+------------------------------------------------------------------+
//| Clase para el acceso a los símbolos                              |
//| en la cadena como en el array de símbolos                        |
//+------------------------------------------------------------------+
class CString
  {
   string            m_string;

public:
                     CString(string str=NULL):m_string(str) { }
   ushort operator[] (int x) { return(StringGetCharacter(m_string,x)); }
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- array para recibir símbolos desde una cadena
   int     x[]={ 19,4,18,19,27,14,15,4,17,0,19,14,17,27,26,28,27,5,14,
```

© 2000-2025, MetaQuotes Ltd.

```
                  17,27,2,11,0,18,18,27,29,30,19,17,8,13,6 };
   CString str("abcdefghijklmnopqrstuvwxyz[ ]CS");
   string  res;
//--- componemos una frase usando símbolos desde la variable str
   for(int i=0,n=ArraySize(x);i<n;i++)
     {
      res+=ShortToString(str[x[i]]);
     }
//--- mostramos resultado
   Print(res);
  }
```

Otro ejemplo de sobrecarga de la operación de indexación es el trabajo con matrices. Una matriz representa un array bidimensional dinámico, los tamaños de los arrays no están definidos de antemano. Por eso no se puede declarar un array de forma array[][] sin especificar el tamaño de la segunda dimensión y luego pasar este array como un parámetro. Como una posible solución puede ser una clase especial CMatrix que contiene un array de los objetos de la clase CRow.

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- operaciones de adición y multiplicación de matrices
   CMatrix A(3),B(3),C();
//--- preparamos arrays para las cadenas
   double a1[3]={1,2,3}, a2[3]={2,3,1}, a3[3]={3,1,2};
   double b1[3]={3,2,1}, b2[3]={1,3,2}, b3[3]={2,1,3};
//--- llenamos matrices
   A[0]=a1; A[1]=a2; A[2]=a3;
   B[0]=b1; B[1]=b2; B[2]=b3;
//--- imprimimos las matrices en el diario "Asesores Expertos"
   Print("---- elementos de la matriz A");
   Print(A.String());
   Print("---- elementos de la matriz B");
   Print(B.String());

//--- suma de matrices
   Print("---- suma de matrices A y B");
   C=A+B;
//--- impresión de la representación string formateada
   Print(C.String());

//--- multiplicación de matrices
   Print("---- multiplicación de matrices A y B");
   C=A*B;
   Print(C.String());

//--- y ahora vamos a mostrar cómo obtener los valores en el estilo de los arrays diná
   Print("Mostramos los valores de la matriz C elemento por elemento");
//--- repasamos en el ciclo las filas de la matriz - objetos CRow
   for(int i=0;i<3;i++)
     {
      string com="| ";
      //--- formamos filas desde la matriz para el valor
      for(int j=0;j<3;j++)
        {
         //--- obtenemos los elementos de la matriz por números de la fila y columna
         double element=C[i][j];// [i] - acceso CRow en el array m_rows[] ,
                                // [j] - operador de indexación sobrecargado en CRow
         com=com+StringFormat("a(%d,%d)=%G ; ",i,j,element);
        }
      com+="|";
      //--- imprimimos el valor de la fila
      Print(com);
     }
  }
//+------------------------------------------------------------------+
//|  Clase "Fila"                                                    |
//+------------------------------------------------------------------+
class CRow
  {
private:
   double            m_array[];
public:
   //--- constructores y destructor
                     CRow(void)           { ArrayResize(m_array,0);     }
                     CRow(const CRow &r)  { this=r;                     }
                     CRow(const double &array[]);
```

© 2000-2025, MetaQuotes Ltd.

```
                    ~CRow(void){};
   //--- número de elementos en la fila
   int               Size(void) const    { return(ArraySize(m_array));}
   //--- devuelve la cadena con valores
   string            String(void) const;
   //--- operador de indexación
   double            operator[](int i) const  { return(m_array[i]);   }
   //--- operadores de asignación
   void              operator=(const double  &array[]); // array
   void              operator=(const CRow & r);         // otro objeto CRow
   double            operator*(const CRow &o);          // objeto CRow para multiplica
  };
//+------------------------------------------------------------------+
//|  Constructor para inicializar una fila con un array              |
//+------------------------------------------------------------------+
void  CRow::CRow(const double &array[])
  {
   int size=ArraySize(array);
//--- si el array no está vacío
   if(size>0)
     {
      ArrayResize(m_array,size);
      //--- llenamos con valores
      for(int i=0;i<size;i++)
         m_array[i]=array[i];
     }
//---
  }
//+------------------------------------------------------------------+
//|  Operación de asignación para array                              |
//+------------------------------------------------------------------+
void CRow::operator=(const double &array[])
  {
   int size=ArraySize(array);
   if(size==0) return;
//--- llenamos array con valores
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//|  Operación de asignación para CRow                                 |
//+------------------------------------------------------------------+
void CRow::operator=(const CRow  &r)
  {
   int size=r.Size();
   if(size==0) return;
//--- llenamos array con valores
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=r[i];
//---
  }
//+------------------------------------------------------------------+
//|  Operador de multiplicación por otra fila                          |
//+------------------------------------------------------------------+
double CRow::operator*(const CRow &o)
  {
   double res=0;
//--- comprobaciones
   int size=Size();
   if(size!=o.Size() || size==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCSIG__,": Error de multiplicación de dos matrices, sus tamaños no co
       return(res);
      }
//--- multiplicamos los arrays elemento por elemento y sumamos los productos
   for(int i=0;i<size;i++)
      res+=m_array[i]*o[i];
//--- resultado
   return(res);
  }
//+------------------------------------------------------------------+
//|  Devuelve la representaión string formateada                     |
//+------------------------------------------------------------------+
string CRow::String(void) const
  {
   string out="";
//--- si el tamaño del array es superior a cero
   int size=ArraySize(m_array);
//--- trabaja sólo con el número de elementos en el array superior a cero
   if(size>0)
     {
      out="{";
      for(int i=0;i<size;i++)
        {
         //--- reunimos los valores en la cadena
         out+=StringFormat(" %G;",m_array[i]);
        }
      out+=" }";
     }
//--- resultado
   return(out);
  }
//+------------------------------------------------------------------+
//|  Clase "Matriz"                                                  |
//+------------------------------------------------------------------+
class CMatrix
  {
private:
   CRow              m_rows[];

public:
   //--- constructores y destructor
                     CMatrix(void);
                     CMatrix(int rows)  { ArrayResize(m_rows,rows);            }
                    ~CMatrix(void){};
   //--- obtener tamaños de la matriz
   int               Rows()       const { return(ArraySize(m_rows));           }
   int               Cols()       const { return(Rows()>0? m_rows[0].Size():0); }
   //--- devuelve los valores de la columna en forma de una fila CRow
   CRow              GetColumnAsRow(const int col_index) const;
   //--- devuelve una cadena con los valores de la matriz
   string            String(void) const;
   //--- operador de indexación devuelve la cadena por su número
   CRow *operator[](int i) const        { return(GetPointer(m_rows[i]));       }
   //--- operador de adición
   CMatrix           operator+(const CMatrix &m);
   //--- operador de multiplicación
   CMatrix           operator*(const CMatrix &m);
   //--- operador de asignación
   CMatrix          *operator=(const CMatrix &m);
  };
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Un constructor predefinido, crea un array de filas con el tamaño cero |
//+------------------------------------------------------------------+
CMatrix::CMatrix(void)
  {
//--- número de filas cero en la matriz
   ArrayResize(m_rows,0);
//---
  }
//+------------------------------------------------------------------+
//|  Devuelve los valores de la columna en forma de la fila CRow                  |
//+------------------------------------------------------------------+
CRow  CMatrix::GetColumnAsRow(const int col_index) const
  {
//--- una variable para recibir valores desde la columna
   CRow row();
//--- número de filas en la matriz
   int rows=Rows();
//--- si el número de filas es mayor a cero, ejecutamos la operación
   if(rows>0)
     {
      //--- array para recibir valores de la columna con el índice col_index
      double array[];
      ArrayResize(array,rows);
      //--- llenando array
      for(int i=0;i<rows;i++)
        {
         //--- comprobación del número de la columna para la fila i para ver si sale f
         if(col_index>=this[i].Size())
           {
            Print(__FUNCSIG__,": ¡Error! El número de la columna es ",col_index,"> del
            break; // row se queda como un objeto no inicializado
           }
         array[i]=this[i][col_index];
        }
      //--- creamos la fila CRow a base de los valores del array
      row=array;
     }
//--- resultado
   return(row);
  }
//+------------------------------------------------------------------+
//|  Suma de dos matrices                                            |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator+(const CMatrix &m)
  {
//--- número de filas y columnas en la matriz pasada
   int cols=m.Cols();
   int rows=m.Rows();
//--- matriz para recibir el resultado de adición
   CMatrix res(rows);
//--- los tamaños de la matriz deben coincidir
   if(cols!=Cols() || rows!=Rows())
     {
      //--- no se puede sumar
      Print(__FUNCSIG__,": Error de adición de dos matrices, los tamaños no coinciden'
      return(res);
     }
//--- array auxiliar
   double arr[];
   ArrayResize(arr,cols);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- repasamos las filas para sumar
   for(int i=0;i<rows;i++)
     {
      //--- escribimos los resultados de adición de las cadenas de las matrices en el
      for(int k=0;k<cols;k++)
        {
         arr[k]=this[i][k]+m[i][k];
        }
      //--- colocamos el array en la fila de la matriz
      res[i]=arr;
     }
//--- devolvemos el resultado de adición de matrices
   return(res);
  }
//+------------------------------------------------------------------+
//|  Multiplicación de dos matrices                                  |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator*(const CMatrix &m)
  {
//--- número de columnas de la primera matriz, número de filas en la matriz pasada
   int cols1=Cols();
   int rows2=m.Rows();
   int rows1=Rows();
   int cols2=m.Cols();
//--- matriz para recibir el resultado de multiplicación
   CMatrix res(rows1);
//--- las matrices tiene que ser compatibles
   if(cols1!=rows2)
     {
      //--- no se puede multiplicar
      Print(__FUNCSIG__,": Error de multiplicación de dos matrices, el formato es inco
            "- el número de columnas en el primer factor debe ser igual al número de f
      return(res);
     }
//--- array auxiliar
   double arr[];
   ArrayResize(arr,cols1);
//--- llenamos filas en la matriz de multiplicación
   for(int i=0;i<rows1;i++)// repasamos filas
     {
      //--- ponemos a cero el array receptor
      ArrayInitialize(arr,0);
      //--- repasamos elementos en la fila
      for(int k=0;k<cols1;k++)
        {
         //--- cogeremos desde la matriz m los valores de la columna k en forma de la
         CRow column=m.GetColumnAsRow(k);
         //--- multiplicamos dos filas y escribimos el resultado de la multiplicación
         arr[k]=this[i]*column;
        }
      //--- colocamos el array arr[] en la fila i de la matriz
      res[i]=arr;
     }
//--- devolvemos el producto de dos matrices
   return(res);
  }
//+------------------------------------------------------------------+
//|  Operación de asignación                                         |
//+------------------------------------------------------------------+
CMatrix *CMatrix::operator=(const CMatrix &m)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- encontramos y fijamos el número de filas
   int rows=m.Rows();
   ArrayResize(m_rows,rows);
//--- llenamos nuestras filas con los valores de las filas de matriz pasada
   for(int i=0;i<rows;i++) this[i]=m[i];
//---
   return(GetPointer(this));
  }
//+------------------------------------------------------------------+
//|  Representación string de la matriz                              |
//+------------------------------------------------------------------+
string CMatrix::String(void) const
  {
   string out="";
   int rows=Rows();
//--- formamos cadena por cadena
   for(int i=0;i<rows;i++)
     {
      out=out+this[i].String()+"\r\n";
     }
//--- resultado
   return(out);
  }
```

Véase también

Sobrecarga, Operaciones aritméticas, Sobrecarga de funciones, Prioridades y orden de las operaciones

© 2000-2025, MetaQuotes Ltd.

Descripción de funciones externas

Las funciones externas, definidas en el otro módulo, deben ser descritas con claridad. La descripción incluye el tipo de valor retornado, el nombre de la función y el conjunto de parámetros de entrada con su tipo. La falta de esta descripción puede llevar a los errores en la fase de compilación, composición o ejecución del programa. Durante la descripción de un objeto externo, use la palabra clave *#import* indicando el módulo.

Ejemplos:

```
#import "user32.dll"
  int     MessageBoxW(int hWnd ,string szText,string szCaption,int nType);
  int     SendMessageW(int hWnd,int Msg,int wParam,int lParam);
#import "lib.ex5"
  double  round(double value);
#import
```

Mediante la importación es muy fácil describir las funciones invocadas de las DLL externas o de las bibliotecas compiladas EX5. Las bibliotecas EX5 son unos archivos ex-5 que tienen la propiedad de library. Sólo las funciones descritas con el modificador export pueden ser importadas de las bibliotecas EX5.

En caso de usar las bibliotecas DLL y EX5 simultáneamente, hay que recordar que tienen que tener diferentes nombres (independientemente de las carpetas de su ubicación). Todas las funciones importadas obtienen el campo de visibilidad correspondiente al "nombre del archivo" de la biblioteca.

Ejemplo:

```
#import "kernel32.dll"
   int GetLastError();
#import "lib.ex5"
   int GetLastError();
#import

class CFoo
  {
public:
   int            GetLastError() { return(12345); }
   void           func()
     {
      Print(GetLastError());           // llamada al método de clase
       Print(::GetLastError());         // llamada a la función MQL5
       Print(kernel32::GetLastError()); // llamada a la función kernel32.dll
       Print(lib::GetLastError());      // llamada a la función lib.ex5
     }
  };

void OnStart()
  {
   CFoo foo;
   foo.func();
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

Véase también

Sobrecarga, Funciones virtuales, Polimorfismo

© 2000-2025, MetaQuotes Ltd.

Exportación de funciones

Hay posibilidad de usar en el programa MQL5 la función que ha sido declarada en otro programa MQL5 con el postmodificador *export*. Esta función se llama exportable, y puede invocarse de otros programas después de la compilación.

```
int Function() export
  {
  }
```

Este modificador indica al compilador que inserte la función en la tabla de funciones EX5 que serán exportadas por el archivo ejecutor ex5. Sólo las funciones con este modificador serán accesibles ("visibles") desde otros programas mql5.

La propiedad library indica al compilador que este archivo EX5 va a ser una biblioteca, y el compilador lo pondrá en la cabecera de EX5.

Todas las funciones planificadas como exportables deben ser marcadas con el modificador *export*.

Véase también

Sobrecarga, Funciones virtuales, Polimorfismo

© 2000-2025, MetaQuotes Ltd.

Funciones de procesamiento de eventos

En el lenguaje MQL5 está previsto el procesamiento de algunos eventos predefinidos. Las funciones que manejan estos eventos tienen que ser definidos en el programa MQL5; el nombre de la función, el tipo de valor devuelto, composición de parámetros (si éstos existen) y sus tipos tienen que corresponder estrictamente a la descripción de la función de procesamiento de eventos.

El manejador de eventos del terminal de cliente identifica las funciones, que procesan uno u otro evento, precisamente por el tipo de valor devuelto y por los tipos de parámetros. La función no se usa para manejar los eventos, si en ésta están indicados otros parámetros que no corresponden a las descripciones siguientes, o está indicado otro tipo de valor devuelto.

OnStart

La función OnStart() es manejador de evento Start, que se genera automáticamente **sólo** para los **scripts** a ser ejecutados. Esta función debe ser del tipo void, sin tener parámetros:

```
void OnStart();
```

Para la función OnStart() se admite especificar el tipo de valor devuelto int.

OnInit

La función OnInit() es el manejador de evento Init. Puede tener el tipo void o int, no tiene parámetros:

```
void OnInit();
```

El evento Init se genera justo después de haberse cargado un Asesor Experto o un indicador; este evento no se genera para los scripts. La función OnInit() se usa para la inicialización. Si OnInit() tiene el valor devuelto del tipo int, entonces el código no nulo de la devolución significa una inicialización fallida y genera el evento Deinit con el código de la causa de deinicialización REASON_INITFAILED.

Si se retorna el valor INIT_FAILED, el asesor será descargado del gráfico forzosamente.

Si se retorna el valor INIT_FAILED, el indicador no será descargado del gráfico. En este caso, además, el asesor que quedará en el gráfico no funcionará: el manejador de eventos en el indicador no será llamado.

Para la optimización de los parámetros de entrada del EA se recomienda utilizar los valores de la enumeración ENUM_INIT_RETCODE como el código de devolución. Estos valores sirven para organizar el control del proceso de optimización, incluyendo la selección de los agentes de pruebas más apropiados. Durante la inicialización del EA, antes de iniciar el mismo proceso de simulación, se puede solicitar la información sobre la configuración y los recursos del agente (número de núcleos, volumen de la memoria libre, etc.) utilizando la función TerminalInfoInteger(). Y basándose en la información recibida permitir el uso de este agente, o bien rechazarlo para la optimización de este EA.

ENUM_INIT_RETCODE

Identificador	Descripción
INIT_SUCCEEDED	La inicialización se ha finalizado con éxito. Se puede seguir con la simulación del EA.

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
	Este código significa lo mismo que el valor nulo - la inicialización del EA en el Probador ha pasado con éxito.
INIT_FAILED	Inicialización fallida. No merece la pena continuar la simulación debido a los errores insuperables. Por ejemplo, no se ha podido crear el indicador necesario para el funcionamiento del EA. La devolución de este valor significa lo mismo que la devolución de un valor distinto del cero. Significa que la inicialización del EA en el Probador no ha tenido éxito.
INIT_PARAMETERS_INCORRECT	Se utiliza por el programador para marcar un conjunto incorrecto de parámetros de entrada. La cadena del resultado que contiene este código de retorno se colorea con el rojo en la tabla general de optimización. La simulación para este conjunto de parámetros del EA no va a llevarse a cabo, el agente está disponible para recibir una nueva tarea. Cuando el Probador de Estrategias recibe este valor, nunca va a pasar esta tarea a otros agentes para que vuelvan a ejecutarlo.
INIT_AGENT_NOT_SUITABLE	Durante la inicialización no ha surgido ningún error en el funcionamiento del programa, pero por alguna razón este agente no vale para hacer la prueba. Por ejemplo, no hay suficiente memoria operativa, no hay soporte OpenCL, etc. Después de la devolución de este código, el agente ya no va a recibir tareas hasta que se finalice esta optimización.

La función OnInit() del tipo void siempre indica a una inicialización exitosa.

OnDeinit

La función OnDeinit() se llama durante la deinicialización y juega papel de manejador de evento Deinit. Tiene que ser declarada con el tipo void y tener un parámetro del tipo const int que contiene el código de la causa de deinicialización. Si está declarado otro tipo, el compilador lanzará un aviso pero la función no será invocada. Para los scripts el evento Deinit no se genera, y por tanto, en éstos no se puede usar la función OnDeinit().

```
void OnDeinit(const int reason);
```

El evento Deinit se genera para los Asesores Expertos e indicadores en las siguientes ocasiones:

- antes de la reinicialización debido al cambio de símbolo o período del gráfico al cual el programa mql5 es atado;
- antes de la reinicialización debido al cambio de los parámetros de entrada;
- antes de descargar un programa mql5.

OnTick

© 2000-2025, MetaQuotes Ltd.

El evento NewTick se genera **únicamente para los Asesores Expertos** cuando se recibe un nuevo tick para el símbolo, al diagrama del cual está atado el Asesor. Es inútil determinar la función OnTick() en un indicador personalizado o en un script, porque para ellos el evento Tick no se genera.

El evento NewTick se genera sólo para los Asesores Expertos pero esto no significa que ellos tienen que tener la función OnTick() de una forma obligatoria, porque para los Asesores se generan no sólo los eventos NewTick, sino también los eventos Timer, BookEvent y ChartEvent. Tiene que ser declarada con el tipo void, no tiene parámetros:

```
void OnTick();
```

OnTimer

La función OnTimer() se llama cuando se inicia el evento Timer que se genera por el temporizador del sistema sólo para los asesores e indicadores; no se puede usarla en los scripts. La frecuencia del inicio de este evento se establece cuando la función EventSetTimer() efectua la suscripción para obtener avisos sobre el evento Timer.

Para dar de baja dicha suscripción a recibir los eventos del temporizador para un Asesor en concreto se utiliza la función EventKillTimer(). La función tiene que ser declarada con el tipo void, no tiene parámetros:

```
void OnTimer();
```

Se recomienda llamar a la función EventSetTimer() una sola vez en la función OnInit(), y también una sola vez EventKillTimer() en OnDeinit().

Cada Asesor Experto y cada indicador trabaja con su propio temporizador y recibe eventos sólo de él. Una vez terminada la sesión del programa mql5, el temporizador se elimina de una manera forzosa, si ha sido creado pero no ha sido desactivado por la función EventKillTimer().

OnTrade

La función se llama cuando se inicia el evento Trade, que aparece si se cambia la lista de las órdenes presentadas y posiciones abiertas, historial de órdenes y historial de transacciones. Cuando cualquier operación comercial (presentación de orden pendiente, apertura/cierre de posición, establecimiento de stops, accionamiento de órdenes pendientes, etc.) se efectua de una manera correspondiente, el historial de órdenes y transacciones y/o la lista de posiciones y órdenes corrientes se cambian.

```
void OnTrade();
```

Al recibir este evento (si esto requieren las condiciones de la estrategia comercial), el mismo usuario debe comprobar en el código el estado de la cuenta. Si la llamada a la función OrderSend() se ha realizado con éxito y ha devuelto el valor true, esto significa que el servidor comercial ha colocado la orden en la cola para ser ejecutado y le ha asignado un número de ticket. En cuanto el servidor procese esta orden, el evento Trade será generado. Y si el usuario ha memorizado el valor del ticket, durante el procesamiento de evento OnTrade() podrá averiguar con su ayuda qué es lo que haya pasado exactamente con la orden.

OnTradeTransaction

Como resultado de ejecución de ciertas acciones con la cuenta de trading su estado se cambia. A estas acciones les pertenecen:

© 2000-2025, MetaQuotes Ltd.

- El envío de una solicitud comercial por parte de cualquier aplicación MQL5 en el terminal de cliente utilizando la función OrderSend y OrderSendAsync, con su posterior ejecución;
- El envío de una solicitud comercial a través de la interfaz gráfica del terminal y su posterior ejecución;
- El accionamiento de órdenes pendientes y órdenes Stop en el servidor;
- La ejecución de operaciones en el servidor de trading.

Como resultado de estas acciones, para la cuenta se ejecutan las transacciones comerciales:

- tramitación de la solicitud comercial;
- cambio de órdenes abiertas;
- cambio del historial de órdenes;
- cambio del historial de operaciones;
- cambio de posiciones.

Por ejemplo, al enviar una orden de compra, ésta se tramita, para la cuenta se crea una orden de compra correspondiente, se realiza la ejecución de la orden, su eliminación de la lista de las abiertas, se agrega al historial de órdenes, luego la operación correspondiente se agrega al historial, y se crea una posición nueva. Pues todas estas acciones son transacciones comerciales. La llegada de cada una de estas transacciones al terminal es el evento TradeTransaction. Este evento llama al manejador OnTradeTransaction

```
void  OnTradeTransaction(
   const MqlTradeTransaction&    trans,        // estructura de transacción comercial
   const MqlTradeRequest&        request,      // estructura de solicitud
   const MqlTradeResult&         result        // estructura de respuesta
   );
```

El manejador contiene tres parámetros:

- **trans** - este parámetro obtiene la estructura MqlTradeTransaction que describe la transacción comercial aplicada a la cuenta de trading;
- **request** - este parámetro obtiene la estructura MqlTradeRequest que describe la solicitud comercial;
- **result** - este parámetro obtiene la estructura MqlTradeResult que describe el resultado de ejecución de la solicitud comercial.

Los dos últimos parámetros **request** y **result** se llenan con los valores sólo para la transacción del tipo TRADE_TRANSACTION_REQUEST, la información sobre el parámetro se puede obtener del parámetro *type* de la variable **trans**. Fíjense que en este caso el campo *request_id* en la variable **result** contiene el identificador de la solicitud comercial **request** cuya ejecución ha provocado la aparición de la transacción comercial descrita en la variable **trans**. La presencia del identificador de la solicitud permite vincular la acción ejecutada (llamada a la función OrderSend o OrderSendAsync) con el resultado de esta acción que se traspasa en OnTradeTransaction().

Una solicitud comercial enviada desde el terminal manualmente o a través de las funciones de trading OrderSend()/OrderSendAsync() puede ocasionar en el servidor de trading varias transacciones consecutivas. Entendiéndose que el orden de llegada de estas transacciones al terminal no se garantiza, por eso no se puede construir su algoritmo de trading esperando la llegada de unas transacciones comerciales tras la llegada de otras.

© 2000-2025, MetaQuotes Ltd.

- Todos los tipos de transacciones comerciales se describen en la enumeración ENUM_TRADE_TRANSACTION_TYPE.
- La estructura MqlTradeTransaction que describe la transacción comercial se llena de una manera diferente en función del tipo de transacción. Por ejemplo, para las transacciones del tipo TRADE_TRANSACTION_REQUEST hay que analizar sólo un campo - type (tipo de la transacción comercial). Para obtener la información adicional hay que analizar el segundo y el tercer parámetro de la función OnTradeTransaction (request y result). Más detalles se puede encontrar en el apartado "Estructura de transacción comercial".
- No toda la información disponible sobre las órdenes, operaciones y posiciones (por ejemplo, un comentario) se pasa en la descripción de una transacción comercial. Para obtener la información más detallada, hay que usar las funciones OrderGet*, HistoryOrderGet*, HistoryDealGet* y PositionGet*.

Una vez aplicadas las transacciones comerciales a la cuenta de cliente, ellas se colocan sucesivamente a la cola de transacciones comerciales del terminal, de donde se pasan sucesivamente al punto de entrada OnTradeTransaction en orden de su llegada al terminal.

Mientras que el EA procese las transacciones comerciales con el manejador OnTradeTransaction, el terminal sigue procesando las transacciones que vayan llegando. De esta manera, el estado de la cuenta de trading ya puede cambiarse durante el proceso de trabajo del OnTradeTransaction. Por ejemplo, mientras que el programa MQL5 esté procesando el evento de agregación de una orden nueva, ésta puede ser ejecutada, eliminada de la lista de las abiertas y pasada al historial. A continuación, el programa será avisado sobre todos estos eventos.

La longitud de la cola de transacciones es de 1024 elementos. Si OnTradeTransaction va a tardar en procesar la transacción de turno, las transacciones antiguas pueden ser expulsadas por las nuevas.

- Por lo general, no existe la proporción exacta respecto al número de llamadas a OnTrade y a OnTradeTransaction. Una llamada a OnTrade corresponde a una o varias llamadas a OnTradeTransaction.
- OnTrade se invoca tras las llamadas correspondientes a OnTradeTransaction.

OnTester

La función OnTester() es el manejador del evento Tester que se genera automáticamente una vez terminado el chequeo histórico del Asesor Experto en el intervalo de datos especificado. La función tiene que ser declarada con el tipo double, no tiene parámetros:

```
double OnTester();
```

La función se invoca justamente antes de la llamada a la función OnDeinit() y tiene el tipo del valor devuelto double. La función OnTester() puede ser usada solamente en los Asesores Expertos durante el chequeo. En primer lugar está destinada para el cálculo de un valor que se usa como criterio Custom max durante la optimización genética de los parámetros de entrada.

Durante la optimización genética la selección de resultados dentro de una generación se realiza en orden descendiente. Es decir, desde el punto de vista del criterio de optimización, los mejores resultados son los que tienen el mayor valor (para el criterio de optimización Custom max se toman en cuenta los valores devueltos por la función OnTester). Con este tipo de selección los peores valores se colocan al final, y luego no toman parte en la formación de la siguiente generación.

© 2000-2025, MetaQuotes Ltd.

OnTesterInit

Se llama en los expertos al suceder el evento TesterInit para el procesamiento de las acciones necesarias antes de comenzar la optimización en el simulador de estrategias. Existen dos variantes de la función.

La versión con retorno del resultado

```
int  OnTesterInit(void);
```

Valor retornado

Valor del tipo int, cero indica que el experto iniciado en el gráfico antes del comienzo de la optimización se ha inicializado con éxito.

Es prioritario el uso de la llamada de OnTesterInit() con retorno del resultado de ejecución, ya que este método permite no solo ejecutar la inicialización del programa, sino también retornar el código de error en el caso de que el programa finalice antes de tiempo. Retorna cualquier valor distinto a INIT_SUCCEEDED (0); indica que existe un error y la optimización no será iniciada.

La versión sin retorno de resultado se ha dejado solo por compatibilidad con los códigos antiguos. No se recomienda su uso

```
void  OnTesterInit(void);
```

El EA que cuenta con el manejador OnTesterDeinit() o OnTesterPass(), al iniciarse la optimización, se carga automáticamente en un gráfico separado con el símbolo y período especificados en el Probador y recibe el evento TesterInit. Esta función se utiliza para inicializar el EA antes del inicio de la optimización para el posterior procesamiento de los resultados de la optimización.

OnTesterPass

La función OnTesterPass() es el manejador del evento TesterPass que se genera automáticamente cuando llega un frame durante la optimización del EA en el Probador de Estrategias. La función tiene que ser definida con el tipo void. No tiene parámetros:

```
void OnTesterPass();
```

El EA con el manejador OnTesterPass() se carga automáticamente en un gráfico nuevo del terminal con el símbolo/período especificados para la simulación, y recibe durante la optimización los eventos TesterPass cuando llegue un frame. La función está destinada para el procesamiento dinámico de los resultados de la optimización directamente "al vuelo", sin esperar su finalización. La agregación de los frames se realiza por la función FrameAdd(), que puede ser invocada cuando se finaliza el repaso único en el manejador OnTester().

OnTesterDeinit

La función OnTesterDeinit() es el manejador del evento TesterDeinit que se genera automáticamente tras finalizarse la optimización del EA en el Probador de Estrategias. La función tiene que ser definida con el tipo void. No tiene parámetros:

```
void OnTesterDeinit();
```

© 2000-2025, MetaQuotes Ltd.

El EA con el manejador TesterDeinit() se carga automáticamente en el gráfico al iniciarse la optimización, y recibe el evento TesterDeinit tras su finalización. Esta función está destinada para el procesamiento final de todos los resultados de la optimización.

OnBookEvent

La función OnBookEvent() es manejador de evento BookEvent. El evento BookEvent se genera sólo para los Asesores e indicadores si se cambia el estado de la profundidad de mercado (Depth of Market). Debe tener el tipo void y un parámetro del tipo string:

```
void OnBookEvent (const string& symbol);
```

Para recibir los eventos BookEvent por cualquier símbolo, es suficiente suscribirse previamente a la recepción de estos eventos para este símbolo mediante la función MarketBookAdd(). Para dar de baja la recepción del evento BookEvent por un símbolo concreto, es necesario llamar a la función MarketBookRelease().

A diferencia de otros eventos, el evento BookEvent es de difusión. Eso significa que si un Asesor Experto se suscribe a la recepción del evento BookEvent a través de la función MarketBookAdd, todos los demás Asesores que tienen el manejador OnBookEvent() van a recibir este evento. Por eso es necesario analizar el nombre del símbolo que se traspasa al manejador en calidad del parámetro *const string& symbol*.

OnChartEvent

OnChartEvent() es manejador del grupo de eventos ChartEvent:

- CHARTEVENT_KEYDOWN — evento de pulsación de teclas del teclado cuando la ventana del gráfico está en el foco;
- CHARTEVENT_MOUSE_MOVE — eventos de mover el ratón y pulsar botones del ratón (si para el gráfico está establecida la propiedad CHART_EVENT_MOUSE_MOVE=true);
- CHARTEVENT_OBJECT_CREATE — evento de creación de un objeto gráfico (si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_CREATE=true);
- CHARTEVENT_OBJECT_CHANGE — evento de cambio de propiedades de un objeto a través del diálogo de propiedades;
- CHARTEVENT_OBJECT_DELETE — evento de eliminación de un objeto gráfico (si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_DELETE=true);
- CHARTEVENT_OBJECT_CLICK — evento de cliqueo con el ratón sobre el gráfico;
- CHARTEVENT_OBJECT_DRAG — evento de movimiento de un objeto gráfico mediante el ratón;
- CHARTEVENT_OBJECT_ENDEDIT — evento del fin de edición de texto en el campo de introducción de un objeto gráfico LabelEdit;
- CHARTEVENT_CHART_CHANGE — evento de modificación del gráfico;
- CHARTEVENT_CUSTOM+n — identificador del evento de usuario donde la n se encuentra en el rango de 0 a 65535.
- CHARTEVENT_CUSTOM_LAST — el último identificador aceptable del evento de usuario (CHARTEVENT_CUSTOM+65535).

La función puede invocarse sólo en los Asesores e indicadores. Tiene que poseer el tipo void y 4 parámetros:

© 2000-2025, MetaQuotes Ltd.

```
void OnChartEvent(const int id,         // identificador de evento
                  const long& lparam,   // parámetro de evento del tipo long
                  const double& dparam, // parámetro de evento del tipo double
                  const string& sparam  // parámetro de evento del tipo string
  );
```

Para cada tipo de evento, los parámetros de entrada de la función OnChartEvent() tienen los determinados valores que son necesarios para procesar este evento. En la tabla de abajo se enumeran los eventos y valores pasados como parámetros.

Evento	Valor del parámetro id	Valor del parámetro lparam	Valor del parámetro dparam	Valor del parámetro sparam
Evento de un teclazo	CHARTEVENT_KEYDOWN	código de la tecla pulsada	Número de pulsaciones de la tecla generadas mientras ésta se mantenía en estado pulsado	Valor literal de la máscara de bits que describe el estatus de las teclas del teclado
Eventos del ratón (si para el gráfico está establecida la propiedad CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	coordinada X	coordinada Y	Valor literal de la máscara de bits que describe el estatus de los botones del ratón
Evento de creación de un objeto gráfico (si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_CREATE=true)	CHARTEVENT_OBJECT_CREATE	—	—	Nombre del objeto gráfico creado
Evento de cambio de propiedades de un objeto a través del diálogo de propiedades	CHARTEVENT_OBJECT_CHANGE	—	—	Nombre del objeto gráfico modificado
Evento de eliminación de un objeto gráfico (si para el	CHARTEVENT_OBJECT_DELETE	—	—	Nombre del objeto gráfico eliminado

© 2000-2025, MetaQuotes Ltd.

Evento	Valor del parámetro id	Valor del parámetro lparam	Valor del parámetro dparam	Valor del parámetro sparam
gráfico está establecida la propiedad CHART_EVENT_OBJECT_DELETE=true)				
Evento de cliquear sobre un gráfico	CHARTEVENT_CLICK	coordenada X	coordenada Y	—
Evento de cliquear sobre un objeto gráfico	CHARTEVENT_OBJECT_CLICK	coordenada X	coordenada Y	Nombre del objeto gráfico en el que ha ocurrido el evento
Evento de mover un objeto gráfico con el ratón	CHARTEVENT_OBJECT_DRAG	—	—	Nombre del objeto gráfico desplazado
Evento del fin de edición del texto en el campo de introducción del objeto gráfico	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nombre del objeto gráfico "Campo de texto" donde se ha finalizado la introducción del texto
Evento de modificación del gráfico	CHARTEVENT_CHART_CHANGE	—	—	—
Identificador del evento de usuario	CHARTEVENT_CUSTOM+N	Valor determinado por la función EventChartCustom()	Valor determinado por la función EventChartCustom()	Valor determinado por la función EventChartCustom()

OnCalculate

La función OnCalculate() se invoca sólo en los indicadores si surge la necesidad de calcular los valores de indicador por evento Calculate. Habitualmente eso ocurre cuando se recibe un nuevo tick por símbolo para el que se calcula el indicador. Además, no es obligatorio que el indicador esté anclado a un gráfico de precio del dicho símbolo.

La función OnCalculate() debe tener el tipo de valor devuelto int. Existen dos posibilidades de definición. No se puede usar las dos opciones de la función dentro un indicador.

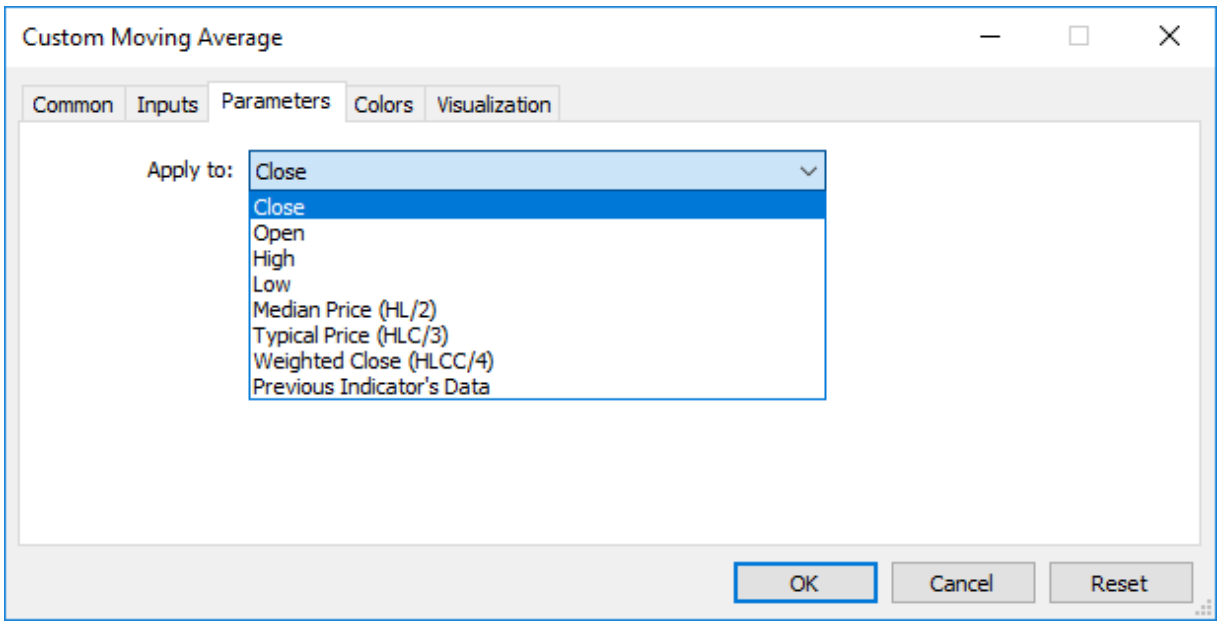
© 2000-2025, MetaQuotes Ltd.

La primera forma de la llamada está destinada para los indicadores que pueden ser calculados sobre un buffer de datos. El ejemplo de este indicador es Custom Moving Average.

```
int OnCalculate (const int rates_total,      // tamaño del array price[]
                 const int prev_calculated,  // cantidad procesada de barras en la ant
                 const int begin,            // de donde se empiezan los datos signif
                 const double& price[]       // array a calcular
   );
```

Como array price[] puede ser pasada una de las series temporales de precios o calculado el búfer de algún indicador. Para determinar la dirección de indexación en el array price[] es necesario invocar la función ArrayGetAsSeries(). Para no depender de los valores predefinidos es necesario invocar incondicionalmente la función ArraySetAsSeries() para los arrays con los que se prevé trabajar.

La elección del indicador o serie temporal apropiados en calidad del array price[] se realiza por el usuario a la hora de iniciar el indicador en la pestaña "Parameters". Para eso se necesita indicar el elemento necesario en la lista desplegable del campo "Apply to".



Par obtener valores del indicador personalizado desde otros programas mql5, se usa la función iCustom() que devuelve el manejador (handle) del indicador para la siguiente operación. Con eso también se puede indicar el array necesario price[] o manejador (handle) de otro indicador. Este parámetro tiene que ser pasado el último en la lista de las variables de entrada del indicador personalizado.

Ejemplo:

```
void OnStart()
  {
//---
   string terminal_path=TerminalInfoString(STATUS_TERMINAL_PATH);
   int handle_customMA=iCustom(Symbol(),PERIOD_CURRENT, "Custom Moving Average",13,0,
   if(handle_customMA>0)
      Print("handle_customMA = ",handle_customMA);
   else
      Print("Cannot open or not EX5 file '"+terminal_path+"\\MQL5\\Indicators\\"+"Cust
  }
```

© 2000-2025, MetaQuotes Ltd.

En este ejemplo el último parámetro pasa el valor PRICE_TYPICAL (desde la enumeración ENUM_APPLIED_PRICE) que indica que el indicador personalizado va a ser construido por los precios típicos recibidos como (High+Low+Close)/3. Si el parámetro no se indica, el indicador se construye por los valores PRICE_CLOSE, es decir, por los precios de cierre de cada barra.

Otro ejemplo que demuestra el traspaso del manejador (handle) de indicador por el último parámetro para especificar el array price[], se muestra en la descripción de la función iCustom().

La segunda forma de la llamada sirve para todos los demás indicadores en los cuales para el cálculo se utiliza más de una serie temporal.

```
int OnCalculate (const int rates_total,      // tamaño de series temporales de entrada
                 const int prev_calculated,  // procesado de barras en la anterior lla
                 const datetime& time[],     // Time
                 const double& open[],       // Open
                 const double& high[],       // High
                 const double& low[],        // Low
                 const double& close[],      // Close
                 const long& tick_volume[],  // Tick Volume
                 const long& volume[],       // Real Volume
                 const int& spread[]         // Spread
   );
```

Los parámetros open[], high[], low[] y close[] contienen los arrays con los precios de apertura, precio máximo, mínimo y precios de cierre del período en curso. El parámetro time[] contiene el array con valores de apertura, el parámetro spread[] es el array que contiene el historial de spreads (si spread está previsto para este instrumento comercial). Los parámetros volume[] y tick_volume[] contienen respectivamente el historial del volumen comercial y del volumen de tick.

Para determinar la dirección de indexación dentro de los arrays time[], open[], high[], low[], close[], tick_volume[], volume[] y spread[] es necesario llamar a la función ArrayGetAsSeries(). Para no depender de los valores predefinidos es necesario invocar incondicionalmente la función ArraySetAsSeries() para los arrays con los que se prevé trabajar.

El primer parámetro rates_total contiene la cantidad de barras disponibles para el indicador para ser calculados, y corresponde a la cantidad de barras disponibles en el gráfico.

Cabe destacar el vínculo entre el valor de la función devuelta OnCalculate() y el segundo parámetro de entrada prev_calculated. Cuando se invoca la función el parámetro prev_calculated contiene el valor que la función OnCalculate() **ha devuelto** durante la llamada **anterior**. Esto permite realizar los algoritmos económicos de cálculo del indicador personalizado con el fin de evitar los cálculos repetidos para las barras que no se hayan cambiado desde el arranque anterior de esta función.

Para eso es suficiente devolver el valor del parámetro rates_total que contiene la cantidad de barras durante la corriente llamada a la función. Si desde la última llamada a la función OnCalculate() los datos de precios han sido cambiados (se ha cargado un historial más profunda o los blancos en el historial han sido llenados), el mismo terminal pondrá el valor cero al parámetro entrante prev_calculated.

Nota: si la función OnCalculate devuelve el valor cero, los valores del indicador no se mostrarán en la ventana DataWindow del terminal de cliente.

Para el mejor entendimiento, será útil iniciar el indicador cuyo código se encuentra más abajo.

Ejemplo de indicador:

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Line
#property indicator_label1  "Line"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDarkBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         LineBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime& time[],
                const double& open[],
                const double& high[],
                const double& low[],
                const double& close[],
                const long& tick_volume[],
                const long& volume[],
                const int& spread[])
  {
//--- obtendremos una cantidad de barras disponibles para el símbolo y período actuale
   int bars=Bars(Symbol(),0);
   Print("Bars = ",bars,", rates_total = ",rates_total,", prev_calculated = ",prev_cal
   Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Véase también

Ejecución del programa, Eventos del terminal de usuario, Trabajo con eventos

© 2000-2025, MetaQuotes Ltd.

Variables

Declaración de variables

Las variables tienen que estar declaradas antes de empezar a usarlas. Para identificar las variables se utilizan los nombres únicos. Las descripciones de variables se utilizan para su determinación y declaración de tipos. Las descripciones no son unos operadores.

Los tipos simples son:

- char, short, int, long, uchar, ushort, uint, ulong - números enteros;
- color - número entero que representa el color RGB;
- datetime - fecha y hora es un número sin signo que contiene la cantidad de segundos pasados desde 0 horas del 1 de enero de 1970;
- bool - valores lógicos true y false;
- double - números de doble precisión en punto flotante;
- float - números de precisión sencilla en punto flotante;
- string - cadenas de caracteres.

Ejemplos:

```
string szInfoBox;
int    nOrders;
double dSymbolPrice;
bool   bLog;
datetime tBegin_Data   = D'2004.01.01 00:00';
color    cModify_Color = C'0x44,0xB9,0xE6';
```

Tipos complejos o compuestos:

Las estructuras son unos tipos de datos compuestos que se construyen con la ayuda de otros tipos.

```
struct MyTime
  {
   int hour;    // 0-23
   int minute;  // 0-59
   int second;  // 0-59
  };
...
MyTime strTime; // variable del tipo de la estructura MyTime declarada anteriormente
```

No se puede declarar las variables del tipo de estructuras hasta que la estructura esté declarada.

Arrays

El array es un conjunto indexado de datos del mismo tipo:

```
int    a[50];       // Array unidimensional de 50 números enteros.
double m[7][50];    // Array bidimensional compuesto por siete arrays,
                    // cada uno de los cuales contiene 50 números.
```

© 2000-2025, MetaQuotes Ltd.

```
MyTime t[100];      // array que contiene elementos del tipo MyTime
```

Sólo un número entero puede ser el índice del array. Se admiten los arrays que no tienen más de cuatro dimensiones. La numeración de los elementos de un array se empieza con 0. El último elemento del array unidimensional tiene el número a 1 menos que el tamaño del array, es decir, la llamada al último elemento del array de 50 números enteros se verá de la siguiente manera a[49]. Lo mismo se puede decir de los arrays multidimensionales, la indexación de una dimensión se realiza de 0 al tamaño de dimensión -1. El último elemento del array bidimensional del ejemplo aparece como m[6][49].

Los arrays estáticos no pueden ser representados como series temporales, es decir, no se les puede aplicar la función ArraySetAsSeries(), que establece el acceso a los elementos del array desde el fin del array a su principio. Si es necesario establecer el acceso al array como en series temporales, hay que usar el objeto del array dinámico.

Al acceder fuera de los límites del array, el subsistema ejecutivo generará un error crítico y la ejecución del programa se detendrá.

Métodos incorporados para trabajar con arrays

Para trabajar con arrays, podemos utilizar las funciones del apartado Operaciones con arrays, así como los métodos incorporados:

Método	Análogo	Descripción
void array.Fill(const scalar value, const int start_pos=0, const int count=-1);	ArrayFill, ArrayInitialize	Rellena un array con el valor indicado
void array.Free();	ArrayFree	Libera el búfer de un array dinámico y establece el tamaño de la dimensión cero como 0 (cero)
int array.Resize(const int range0_size, const int reserve); int array.Resize(const int range_sizes[], const int reserve);	ArrayResize	Establece un nuevo tamaño en la primera dimensión de un array
int array.Print();	ArrayPrint	Muestra en el diario el valor de un array de tipo simple
int array.Size(const int range=-1);	ArraySize, ArrayRange	Retorna el número de elementos en el array completo (rango=-1) o en la dimensión de array especificada
bool array.IsDynamic();	ArrayIsDynamic	Comprueba si un array es dinámico
bool array.IsIndicatorBuffer();		Comprueba si un array es búfer de indicador
bool array.IsSeries();	ArrayIsSeries	Comprueba si un array es una serie temporal

© 2000-2025, MetaQuotes Ltd.

Método	Análogo	Descripción
bool array.AsSeries();	ArrayGetAsSeries	Comprueba la dirección de indexación de un array
bool array.AsSeries(const bool as_series);	ArraySetAsSeries	Establece la dirección de indexación en un array
int array.Copy(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCopy	Copia el valor de un array en otro array
int array.Compare(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCompare	Retorna el resultado de la comparación de dos arrays de tipos simples o dos estructuras de usuario
int array.Insert(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayInsert	Inserta en la matriz-receptor el número indicado de elementos, comenzando por el índice establecido
int array.Remove(const int start_pos, const int count);	ArrayRemove	Elimina de un array el número indicado de elementos comenzando por el índice especificado
int array.Reverse(const int start_pos, const int count);	ArrayReverse	Expande el número especificado de elementos en el array, comenzando por el índice especificado
bool array.Swap(array& arr[]);	ArraySwap	Intercambia contenido con otro array dinámico del mismo tipo
void array.Sort(sort_function);	ArraySort	Clasificación de arrays numéricos según la primera dimensión
int array.Search(scalar value, search_function);	ArrayBsearch	Retorna el índice del primer elemento encontrado en la primera dimensión de un array
int array.Find((scalar value, search_function);		Busca un array usando la función transmitida y retorna el índice del primer elemento encontrado.
array array.Select(scalar value, search_function);		Busca un array usando la función transmitida y retorna un array con todos los elementos encontrados

Especificadores de acceso

Los especificadores de acceso indican al compilador cómo se puede realizar el acceso a las variables, elementos de las estructuras o de las clases.

© 2000-2025, MetaQuotes Ltd.

El especificador const declara una variable como una constante y no permite cambiar el valor de esta variable durante el proceso de ejecución del programa. La inicialización de la variable durante su declaración se permite sólo una vez.

Ejemplo

```
int OnCalculate (const int rates_total,      // tamaño del array price[]
                 const int prev_calculated,  // barras procesadas durante la llamada a
                 const int begin,            // de donde se empiezan los datos signifi
                 const double& price[]       // array para el cálculo
   );
```

Para acceder a los elementos de las estructuras y clases se utilizan los siguientes especificadores:

- public - no limita el acceso con nada a la variable o al método de clase;
- protected - permite el acceso desde los métodos de la misma clase, y también desde los métodos de las clases que se heredan abiertamente. Otro acceso es imposible;
- private - permite el acceso a las variables y métodos de clase únicamente desde los métodos de la misma clase.
- virtual - es aplicable sólo a los métodos de clase (en ningún caso a los métodos de estructuras), y comunica al compilador que este método tiene que ser colocado en la tabla de las funciones virtuales de la clase.

Clases de memoria

Existen tres clases de memoria: static, input y extern. Estos modificadores de la clase de memoria indican al compilador de una manera explícita que las variables correspondientes se distribuyen en la zona de memoria predeterminada que se llama el pool global. Además, estos modificadores indican un procesamiento específico de los datos de variables.

Si una variable declarada a nivel local no es estática, entonces la distribución de la memoria para esta variable se realiza de una manera automática en la pila (stack) de programa. La liberación de memoria destinada para un array no estático también se realiza de forma automática cuando se sale fuera de los límites de visibilidad del bloque, en el cual este array ha sido declarado.

Véase también

Tipos de datos, Encapsulación y extensión de tipos, Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos, Miembros estáticos de la clase

© 2000-2025, MetaQuotes Ltd.

Variables locales

Una variable declarada dentro de una función función es local. La zona de visibilidad de la variable local está limitado por los márgenes de la función dentro de la cual ésta esta declarada. La variable local puede ser inicializada con la ayuda de cualquier expresión. La inicialización de la variable local se realiza cada vez durante la llamada a la función correspondiente. Las variables locales se ubican en zona temporal de la memoria de las función correspondiente.

Ejemplo:

```
int somefunc()
  {
   int ret_code=0;
   ...
   return(ret_code);
  }
```

La zona de aplicación(o visibilidad) de una variable es una parte del programa en la cual se puede referirse a la variable. Las variables declaradas dentro del bloque (a nivel local) van a tener el bloque como su zona de aplicación. La zona de aplicación del bloque se empieza con la declaración de la variable y se termina con la llave derecha final.

Las variables locales declaradas al principio de la función también tienen la zona de aplicación del bloque igual que los parámetros de la función, que también son variables locales. Cualquier bloque puede contener las declaraciones de las variables. Si los bloques están anidados y el nombre de Identificador del bloque exterior coincide con el del bloque interior, el identificador del bloque exterior está "invisible" (oculto) hasta que se termine el trabajo en el bloque interior.

Ejemplo:

```
void OnStart()
  {
//---
   int i=5;       // variable local de la función
     {
      int i=10;  // variable de la función
      Print("En el bloque i = ",i);    // resultado  i = 10;
     }
   Print("Fuera del bloque i = ",i);  // resultado  i = 5;
  }
```

Eso significa que mientras se ejecuta el bloque interno, él ve los valores de sus propios identificadores locales, y no los valores de los identificadores con los mismos nombres en el bloque exterior.

Ejemplo:

```
void OnStart()
  {
//---
   int i=5;       // variable local de la función
   for(int i=0;i<3;i++)
      Print("Dentro for i =",i);
   Print("Fuera del bloque i =",i);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
/* Resultado de ejecución
Dentro for i = 0
Dentro for i = 1
Dentro for i = 2
Fuera del bloque i = 5
*/
```

Las variables locales declaradas como static, tienen la visibilidad del bloque, a pesar de que existan desde el principio de ejecución del programa.

Pila

En cada programa MQL5 para guardar las funciones variables locales que se crean automáticamente se asigna una zona de memoria especial que se llama la pila. Una pila se asigna para todas las funciones, y por defecto su tamaño es de 1Mb. En los expertos y scripts, es posible cambiar el tamaño de la pila con la ayuda de la directiva del compilador #property stacksize (establece el tamaño de la pila en bytes), por defecto, a estos se les asigna un tamaño de 8 Mb.

Las variables locales estáticas se ubican en el mismo lugar que las demás variables estáticas y globales, en una zona especial de la memoria que existe separadamente de la pila. Las variables creadas dinámicamente también utilizan una zona de memoria distinta de la pila.

Con cada llamada a la función, para las variables internas no estáticas se les asigna una zona en la pila. Después de salir de la función, la memoria se queda disponible para su volver a usarse.

Si desde la primera función se hace la llamada a la segunda, ésta en su lugar ocupa en la pila un volumen necesario para sus variables en la memoria restante de la pila. De esta manera, cuando se utilizan las funciones incluidas, la memoria de la pila será ocupada consecutivamente por cada función. Esto puede provocar la falta de memoria durante la siguiente llamada a la función. Esta situación se llama sobrellenado de memoria.

Por eso es mejor utilizar la memoria dinámica para los grandes datos locales - cuando se entra en la función, asignar la memoria necesaria para uso local en el sistema (new, ArrayResize()), y cuando se sale de la función, liberar la memoria (delete, ArrayFree()).

Véase también

Tipos de datos, Encapsulación y extensión de tipos, Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Parámetros formales

Los parámetros pasados a la función son locales. Su zona de visibilidad es el bloque de la función. Los parámetros formales tienen que tener nombres distintos que las variables externas y variables locales definidas dentro de la función. En el bloque de la función a los parámetros formales se les puede asignar algunos valores. Si el parámetro formal está declarado con el modificador const, su valor no se puede cambiar dentro de la función.

Ejemplo:

```
void func(const int & x[], double y, bool z)
  {
   if(y>0.0 && !z)
      Print(x[0]);
   ...
  }
```

Los parámetros formales pueden ser inicializados por las constantes. En este caso, el valor inicializado se considera como el valor por defecto. Los parámetros que siguen después del parámetros inicializado también tienen que ser inicializado.

Ejemplo:

```
void func(int x, double y = 0.0, bool z = true)
  {
   ...
  }
```

Cuando se llama una función como esa, se puede omitir los parámetros inicializados, a su vez serán puestos los valores por defecto.

Ejemplo:

```
func(123, 0.5);
```

Los parámetros de tipos simples se traspasan por valor, es decir, los cambios de la correspondiente variable local de este tipo dentro de la función llamada no se reflejará de ninguna manera en la función llamadora. Los arrays de cualquier tipo y los datos del tipo de estructuras siempre se pasan por referencia. Si es preciso prohibir el cambio del array o el contenido de la estructura, los parámetros de estos tipos tienen que declararse con la palabra clave const.

También existe la posibilidad de pasar por referencia los parámetros de tipos simples. En este caso, la modificación de estos parámetros se reflejará en las correspondientes variables en la función llamada, que han sido pasadas por referencia. Para especificar que un parámetro se pasa por referencia, después del tipo de datos hay que poner el modificador &.

Ejemplo:

```
void func(int& x, double& y, double & z[])
  {
   double calculated_tp;
   ...
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<OrdersTotal(); i++)
     {
      if(i==ArraySize(z))        break;
      if(OrderSelect(i)==false) break;
      z[i]=OrderOpenPrice();
     }
   x=i;
   y=calculated_tp;
  }
```

No se puede inicializar los parámetros pasados por referencia con valores por defecto.

No se puede pasar a la función más de 64 parámetros.

Véase también

Variables Input, Tipos de datos, Encapsulación y extensión de tipos, Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Variables estáticas

La clase de memoria static define una variable estática. El modificador static se indica antes del tipo de datos.

Ejemplo:

```
int somefunc()
  {
   static int flag=10;
   ...
   return(flag);
  }
```

Una variable estática puede ser inicializada por una constante correspondiente a su tipo o por una expresión constante, a diferencia de una variable local simple, que puede ser inicializada por cualquier expresión.

Las variables estáticas existen a partir del momento de ejecución del programa y son inicializadas sólo una vez antes de llamar a la función especializada OnInit(). Si los valores iniciales no están especificados, entonces las variables de la clase estática de la memoria adquieren los valores iniciales cero.

Las variables locales declaradas con la palabra clave static conservan sus valores durante todo el tiempo de vida de la función. Con cada siguiente llamada a la función, estas variables locales ya contienen los valores que tenían durante la llamada anterior.

Cualquier variable en el bloque, salvo los parámetros formales de una función, puede ser definida como estática.

Ejemplo:

```
int Counter()
  {
   static int count;
   count++;
   if(count%100==0) Print("La función Counter ha sido llamada ya ",count," veces");
   return count;
  }
void OnStart()
  {
//---
   int c=345;
   for(int i=0;i<1000;i++)
     {
      int c=Counter();
     }
   Print("c = ",c);
  }
```

Véase también

© 2000-2025, MetaQuotes Ltd.

Tipos de datos, Encapsulación y extensión de tipos, Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos, Miembros estáticos de la clase

© 2000-2025, MetaQuotes Ltd.

Variables globales

Las variables globales se creen mediante la colocación de sus declaraciones fuera de la descripción de una función. Las variables globales se definen al mismo nivel que las funciones, es decir, no son locales en ningún bloque.

Ejemplo:

```
int GlobalFlag=10;   // variable global
int OnStart()
  {
   ...
  }
```

La visibilidad de las variables globales es el programa entero, y están disponibles desde todas las funciones definidas en el programa. Éstas se inicializan por cero, si no está definido explícitamente otro valor inicial. Una variable global puede ser inicializada sólo por una constante correspondiente a su tipo o por una expresión constante.

Global variables are initialized only once after the program is loaded into the client terminal memory and before the first handling of the Init event. For global variables representing class objects, during their initialization the corresponding constructors are called. In scripts global variables are initialized before handling the Start event.

Nota: no se puede confundir las variables declaradas a nivel global con las variables globales del terminal de cliente, el acceso a las cuales se consigue a través de la función GlobalVariable...().

Véase también

Tipos de datos, Encapsulación y extensión de tipos, Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Variables Input

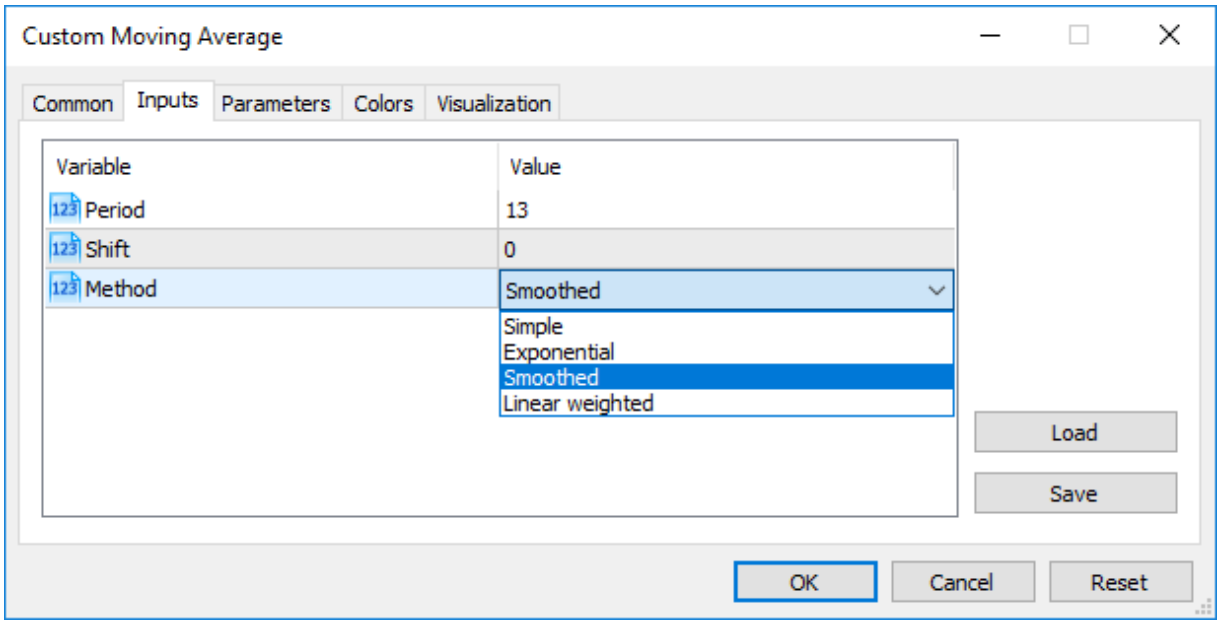
La clase de memoria input define una variable externa. El modificador input se indica antes del tipo de datos. No se puede cambiar el valor de una variable con el modificador input dentro del programa mql5, estas variables son exclusivamente para la lectura. Sólo el usuario puede modificar los valores de las variables input desde la ventana de propiedades de su programa. Las variables externas siempre se reinicializan inmediatamente antes de invocar OnInit().

O comprimento máximo de um nome de variável input é 63 caracteres. Além disso, para um parâmetro de entrada do tipo string o comprimento do valor **máximo** (comprimento da string) pode variar de 191 a 253 caracteres (veja a Observação) O comprimento mínimo é de 0 caracteres (nenhum valor especificado).

Ejemplo:

```
//--- input parameters
input int              MA_Period=13;
input int              MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMMA;
```

Las variables Input determinan los parámetros de entrada del programa, están disponibles desde la ventana Propiedades de la aplicación.



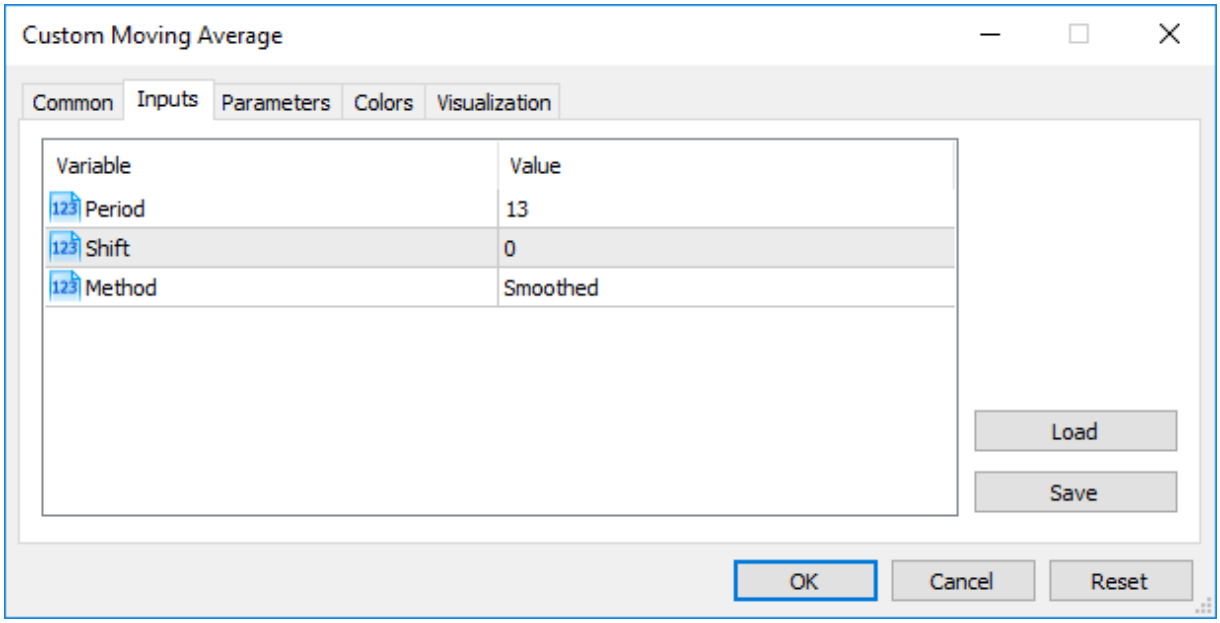
Es posible establecer otro modo de mostrar los nombres de parámetros de entrada en la pestaña "Inputs". Para eso se utiliza un comentario de cadena, el que tiene que ir después de la descripción del parámetro de entrada en la misma cadena. De esa manera, se puede asignar a los parámetros de entrada los nombres más comprensibles para el usuario.

Ejemplo:

```
//--- input parameters
input int              InpMAPeriod=13;         // Smoothing period
```

© 2000-2025, MetaQuotes Ltd.

```
input int             InpMAShift=0;           // Line horizontal shift
input ENUM_MA_METHOD InpMAMethod=MODE_SMMA;  // Smoothing method
```



Nota: Los arrays y variables de tipos compuestos no pueden actuar como las variables input.

Nota: la longitud del comentario de línea para las variables Input no debe superar los 63 caracteres.

Nota: Para las variables input del tipo string, la limitación en la longitud del valor (longitud de la línea) se establece de la forma siguiente:

- el valor del parámetro se representa como línea "nombre_del_parámetro=valor_del_parámetro"(el símbolo '=' se cuenta),
- la longitud máxima de la representación es de 255 caracteres (*total_length_max*=255 o 254 caracteres sin tener en cuenta '='),
- la longitud máxima del nombre del parámetro de línea *parameter_name_length* = 63 caracteres.

Por consiguiente, el tamaño máximo de cadena para un parámetro string se puede calcular mediante la fórmula:

```
parameter_value_length=total_length_max-parameter_name_length=254-parameter_name_lengt
```

Esto ofrece un tamaño máximo de línea de 191 (parameter_name_length=63) a 253 caracteres (parameter_name_length=1).

Traspaso de parámetros al llamar a los indicadores personalizados desde los programas mql5

Los indicadores personalizados se invocan con ayuda de la función iCustom(). Allí, después del nombre del indicador personalizado, deben ir los parámetros de acuerdo estricto con la declaración de variables input de este indicador de usuario. Si se indican menos parámetros que las variables input declaradas en el indicador personalizado invocado, entonces los parámetros que faltan se llenan con los valores especificados durante la declaración de variables.

© 2000-2025, MetaQuotes Ltd.

Si en indicador personalizado se usa la función OnCalculate del primer tipo (es decir, el indicador se calcula en el mismo array de datos), entonces uno de los valores ENUM_APPLIED_PRICE o manejador (handle) del otro indicador debe actuar como el último parámetro durante la llamada de este indicador de usuario. Entonces, todos los parámetros correspondientes a las variables input tienen que ser claramente indicados.

Enumeraciones como parámetro input

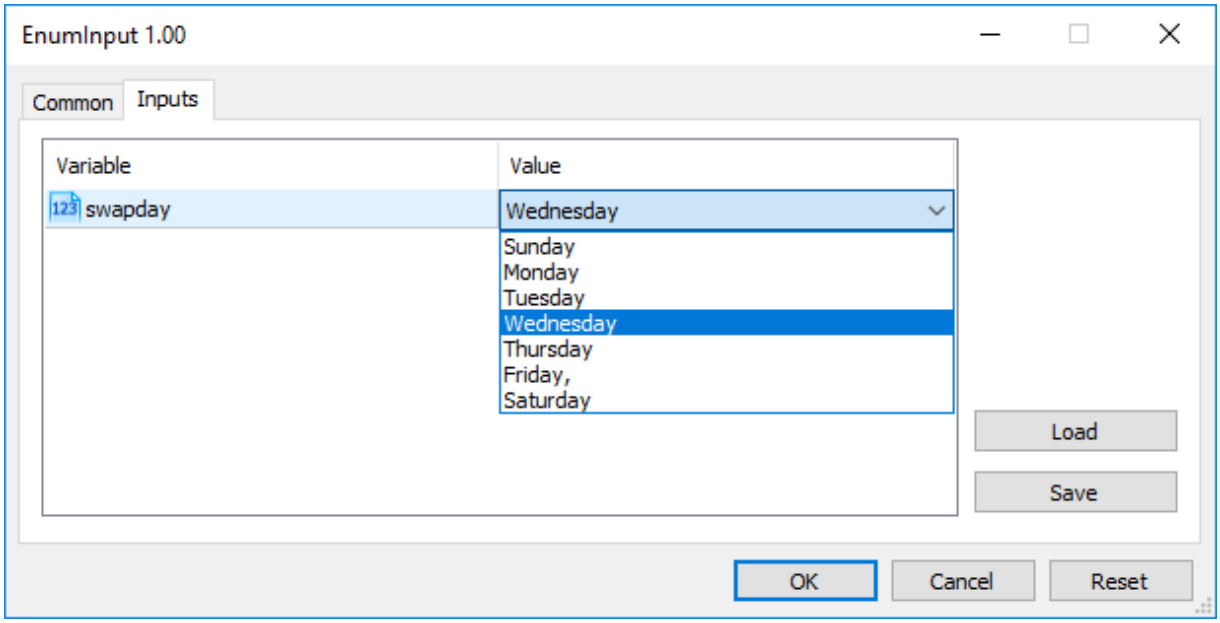
No sólo las enumeraciones "built-in", previstas en el lenguaje MQL5, pueden ser utilizadas como las variables input (parámetros de entrada para los programas mql5), sino las enumeraciones definidas por el usuario. Por ejemplo, podemos crear la enumeración dayOfWeek que describe los días de la semana, y usar la variable input para indicar un día concreto de la semana no como un número, sino en forma más común para el usuario.

Ejemplo:

```
#property script_show_inputs
//--- day of week
enum dayOfWeek
  {
   S=0,      // Sunday
   M=1,      // Monday
   T=2,      // Tuesday
   W=3,      // Wednesday
   Th=4,     // Thursday
   Fr=5,     // Friday,
   St=6,     // Saturday
  };
//--- input parameters
input dayOfWeek swapday=W;
```

Para que el usuario pueda eligir el valor necesario de la ventana de Propiedades durante el inicio del script, usamos el comando de preprocesador #property script_show_inputs. Iniciamos el script y podemos eligir de la lista uno de los valores de la enumeración dayOfWeek. Iniciamos el script EnumInInput y pasamos a la pestaña "Parámetros. Por defecto, el valor del parámetro swapday (día de cómputo del triple swap) es miércoles (W=3), pero nosotros podemos especificar cualquier otro valor y utilizar este valor para cambiar la operación del programa.

© 2000-2025, MetaQuotes Ltd.



La cantidad de posibles valores de enumeración está limitada. Por eso para eligir un valor de entrada se utiliza una lista desplegable. Los nombres mnemotécnicos de los elementos de enumeración se usan en calidad de los valores mostrados dentro de la lista. Pero si un nombre mnemotécnico está asociado con un comentario, como se muestra en el ejemplo, entonces en vez del nombre mnemotécnico se usa el contenido del comentario.

Cada valor de la enumeración dayOfWeek tiene su propio valor de 0 a 6, pero en la lista de parámetros se mostrarán los comentarios indicados para cada valor. Eso da una flexibilidad adicional para escribir los programas con las descripciones claras de los parámetros de entrada.

Cada valor de la enumeración dayOfWeek tiene su valor de 0 a 6, pero en la lista de parámetros serán mostrados los cometarios especificados para cada valor. Esto oporta la flexibilidad adicional para escribir un programa con las descripciones claras de los parámetros de entrada.

Variables con modificador sinput

Las variables con modificador input no sólo permiten establecer los valores de parámetros externos al iniciar los programas, sino también juegan un papel muy importante durante la optimización de estrategias comerciales en el probador. Cada variable input declarada en el EA, salvo el tipo string, puede participar en la optimización.

En algunas ocasiones resulta necesario excluir algunos parámetros externos del programa de la formación de las áreas de posibles pasos en el probador. Precisamente para estas ocasiones existe el modificador de memoria sinput. sinput es la sigla de declaración de una variable estática externa: sinput = static input. Es decir, esta declaración en el código del EA

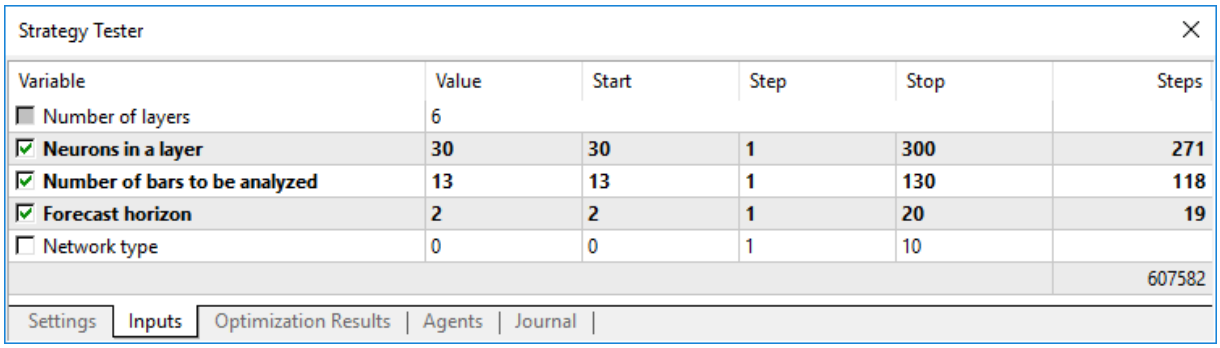
```
sinput       int layers=6;   // Number of layers
```

equivale a la declaración completa

```
static input int layers=6;   // Number of layers
```

© 2000-2025, MetaQuotes Ltd.

Una variable declarada con el modificador sinput es un parámetro de entrada del programa MQL5, y el valor de este parámetro se puede cambiar durante el arranque. Pero esta variable no participa en el proceso de optimización de los parámetros de entrada, es decir, no se efectúa el repaso de sus valores durante la búsqueda del mejor conjunto de parámetros según el criterio establecido.



En la imagen se muestra que el EA tiene 5 parámetros externos, de los cuales el parámetro "Número de capas" ha sido declarado como sinput y es igual a 6. Este parámetro no puede cambiarse durante el proceso de optimización de la estrategia de trading. Sólo se puede establecer para él un valor necesario que va a utilizarse. Los campos Inicio, Paso y Stop para esta variable no están disponibles para poner sus valores.

De esta manera, si establecemos el modificador sinput para una variable, el usuario tiene prohibido optimizar este parámetro. Eso significa que en el Probador de Estrategias el usuario no podrá establecer el valor inicial y final para esta variable con el fin del repaso automático dentro del rango establecido durante el proceso de optimización.

Pero hay una excepción en esta regla: las variables sinput se puede variar en las tareas de optimización utilizando la función ParameterSetRange(). Esta función ha sido creada especialmente para el control programado del área de valores disponibles para cualquier variable input, incluyendo las que han sido declaradas como static input (sinput). Otra función ParameterGetRange() permite obtener los valores de las variables input cuando se inicia la optimización (en el manejador OnTesterInit()), y en caso de necesidad redefinir el paso de cambio y el rango dentro del cual va a repasarse el valor del parámetro a optimizar.

De esta manera, la combinación del modificador sinput con otras dos funciones para el trabajo con los parámetros input permite crear unas reglas sumamente flexibles para establecer los intervalos de optimización de unas variables input dependiendo del valor de otras variables input.

Grupo de parámetros de entrada

Para que resulte cómodo trabajar con los programas de MQL5, podemos dividir los parámetros de entrada en bloques con nombre propio con la ayuda de la palabra clave group. Esto permitirá separar visualmente unos parámetros de otros usando como base la lógica implementada en ellos.

```
input group           "Nombre del grupo"
input int             variable1 = ...
input double          variable2 = ...
input double          variable3= ...
```

© 2000-2025, MetaQuotes Ltd.

Después de dicha declaración, todos los parámetros de entrada se combinan visualmente en el grupo indicado, lo que permite simplificar para el usuario de un programa MQL5 la configuración de los parámtros al iniciar en un gráfico o el simulador de estrategias. La indicación de cada grupo será válida hasta que se declare un nuevo grupo:

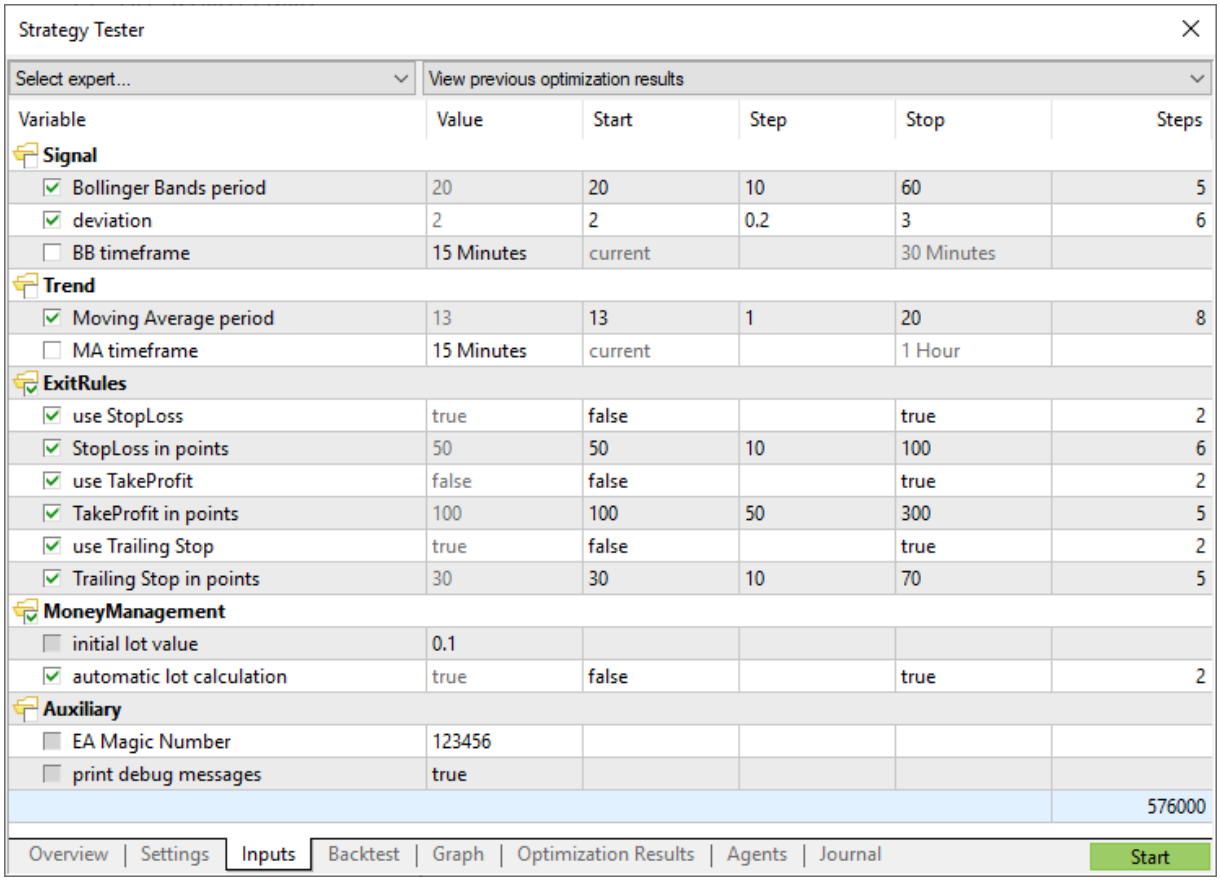
```
input group           "Nombre del grupo #1"
input int             group1_var1 = ...
input double          group1_var2 = ...
input double          group1_var3 = ...

input group           "Nombre del grupo #2
input int             group2_var1 = ...
input double          group2_var2 = ...
input double          group2_var3 = ...
```

Ejemplo de asesor cuyo bloque de parámetros de entrada está dividido según su cometido:

```
input group           "Signal"
input int             ExtBBPeriod   = 20;       // Bollinger Bands period
input double          ExtBBDeviation= 2.0;      // deviation
input ENUM_TIMEFRAMES ExtSignalTF=PERIOD_M15;   // BB timeframe

input group           "Trend"
input int             ExtMAPeriod   = 13;       // Moving Average period
input ENUM_TIMEFRAMES ExtTrendTF=PERIOD_M15;    // MA timeframe

input group           "ExitRules"
input bool            ExtUseSL      = true;     // use StopLoss
input int             Ext_SL_Points = 50;       // StopLoss in points
input bool            ExtUseTP      = false;    // use TakeProfit
input int             Ext_TP_Points = 100;      // TakeProfit in points
input bool            ExtUseTS      = true;     // use Trailing Stop
input int             Ext_TS_Points = 30;       // Trailing Stop in points

input group           "MoneyManagement"
sinput double         ExtInitialLot = 0.1;      // initial lot value
input bool            ExtUseAutoLot = true;     // automatic lot calculation

input group           "Auxiliary"
sinput int            ExtMagicNumber = 123456;  // EA Magic Number
sinput bool           ExtDebugMessage= true;    // print debug messages
```

Al iniciar un asesor de este tipo, en el simulador de estrategias aparece la posibilidad de minimizar y desplegar el bloque de parámetros de entrada de un grupo haciendo doble clic sobre el nombre del mismo, y también de seleccionar todos parámetros de un grupo para la optimización clicando sobre la casilla de verificación de este.

© 2000-2025, MetaQuotes Ltd.



Véase también

iCustom, Enumeraciones, Propiedades de programas

© 2000-2025, MetaQuotes Ltd.

Variables Extern

La palabra clave extern se utiliza para declarar los identificadores de las variables como los identificadores de la clase estática de memoria con el tiempo de vida global. Estas variables existen desde el momento del inicio de programa, y la memoria para ellas se asigna y se inicializa inmediatamente después del inicio del programa.

Se puede crear programas compuestos de varios archivos iniciales. En este caso, para el preprocesador se utiliza el comando #include. Las variables declaradas como extern con el mismo tipo e identificador, pueden existir en diferentes archivos iniciales de un proyecto.

Durante la compilación de todo el proyecto, todas las variables extern con el mismo tipo e identificador, se asocian con una parte de memoria del pool de las variables globales. Las variables extern son útiles para una compilación separada de los archivos iniciales. Las variables extern pueden ser inicializadas, aunque sólo una vez. Está prohibida la existencia de varias variables inicializadas extern del mismo tipo y con el mismo identificador.

Véase también

Tipos de datos, Encapsulación y extensión de tipos, Inicialización de variables, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Inicialización de variables

Cualquier variable puede ser inicializada durante la definición. Si la variable no es inicializada explícitamente, el valor almacenado en esta variable puede ser cualesquiera. La inicialización implícita no se realiza.

Las variables globales y estáticas pueden ser inicializadas sólo por la constante del tipo correspondiente o por una expresión constante. Las variables locales pueden ser inicializadas por cualquier expresión, y no sólo por una constante.

La inicialización de las variables globales y estáticas se realiza sólo una vez. La inicialización de las variables locales se realiza cada vez al llamar a la función correspondiente.

Ejemplos:

```
int    n       = 1;
string s       = "hello";
double f[]     = { 0.0, 0.236, 0.382, 0.5, 0.618, 1.0 };
int    a[4][4] = { {1, 1, 1, 1}, {2, 2, 2, 2}, {3, 3, 3, 3}, {4, 4, 4, 4 } };
//--- de tetris
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                 WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
//--- inicialización de todos los campos de la estructura con valores cero
MqlTradeRequest request={};
```

La lista de valores de los elementos del array tiene que encerrarse dentro de las llaves. Las sucesiones inicializadoras perdidas se consideran iguales a 0.

Si el tamaño del array inicializado no está especificado, éste es determinado por el compilador basándose en el tamaño de la sucesión inicializadora.

Ejemplos:

```
struct str3
  {
   int               low_part;
   int               high_part;
  };
struct str10
  {
   str3              s3;
   double            d1[10];
   int               i3;
  };
void OnStart()
  {
   str10 s10_1={{1,0},{1.0,2.1,3.2,4.4,5.3,6.1,7.8,8.7,9.2,10.0},100};
   str10 s10_2={{1,0},{},100};
   str10 s10_3={{1,0},{1.0}};
//---
   Print("1.  s10_1.d1[5] = ",s10_1.d1[5]);
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("2.  s10_2.d1[5] = ",s10_2.d1[5]);
   Print("3.  s10_3.d1[5] = ",s10_3.d1[5]);
   Print("4.  s10_3.d1[0] = ",s10_3.d1[0]);
  }
```

Para las variables del tipo de estructuras se permite una inicialización parcial, lo mismo se refiere a los arrays estáticos (con un tamaño fijado explícitamente). Se puede inicializar uno o varios primeros elementos de la estructura o array, el resto de los elementos en este caso serán inicializados con ceros.

Véase también

Tipos de datos, Encapsulación y extensión de tipos, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Visibilidad y tiempo de vida de variables

Hay dos tipos principales de zona de visibilidad: zona local de visibilidad y zona global de visibilidad.

Una variable declarada fuera de todas las funciones se coloca en la visibilidad global. Estas variables pueden ser accesibles desde cualquier parte del programa. Estas variables se ubican en el pool global de la memoria, por eso el tiempo de sus vidas coincide con el tiempo de vida del programa.

Una variable declarada dentro del bloque (parte del código encerrada dentro de las llaves) pertenece a la visibilidad local. Esta variable no es visible (por tanto no es accesible) fuera del bloque en el que está declarada. El caso más común de la declaración local es una variable declarada dentro de la función. Una variable con declaración local se coloca en una pila, y su tiempo de vida coincide con el tiempo de vida de la función.

Ya que la zona de visibilidad de una variable local es el bloque donde está declarada, hay posibilidad de declarar las variables con los nombres que coincidan con los de las variables declaradas en otros bloques, y también declaradas en los niveles superiores, así sucesivamente hasta el nivel global.

Ejemplo:

```
void CalculateLWMA(int rates_total,int prev_calculated,int begin,const double &price[]
  {
   int        i,limit;
   static int weightsum=0;
   double     sum=0;
//---
   if(prev_calculated==0)
     {
      limit=MA_Period+begin;
      //--- set empty value for first limit bars
      for(i=0; i<limit; i++) LineBuffer[i]=0.0;
      //--- calculate first visible value
      double firstValue=0;
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
      firstValue/=(double)weightsum;
      LineBuffer[limit-1]=firstValue;
     }
   else
     {
      limit=prev_calculated-1;
     }

   for(i=limit;i<rates_total;i++)
     {
      sum=0;
```

© 2000-2025, MetaQuotes Ltd.

```
      for(int j=0; j<MA_Period; j++) sum+=(MA_Period-j)*price[i-j];
      LineBuffer[i]=sum/weightsum;
     }
//---
  }
```

Preste la atención a la variable i declarada en línea

```
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
```

Su visibilidad es sólo el ciclo for, fuera de este ciclo hay otra variable con el mismo nombre que ha sido declarada al principio de la función. Además, en el cuerpo del ciclo está declarada la variable K, cuya zona de visibilidad es el cuerpo del ciclo.

Se puede declarar las variables locales con el especificador de acceso static. En este caso, el compilador coloca esta variable en el pool global de la memoria. Por tanto, el tiempo de vida de una variable estática coincide con el tiempo de vida del programa. No obstante, la zona de visibilidad de esta variable se limita por el bloque donde está declarada.

Véase también

Tipos de datos, Encapsulación y extensión de tipos, Inicialización de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Creación y eliminación de objetos

Una vez cargado el programa mql5 para ser ejecutado, a cada constante se le asigna una parte de memoria de acuerdo con el tipo de la variable. Dependiendo del nivel de acceso las variables se dividen en dos tipos: variables globales y variables locales, y también dependiendo de las clases de memoria: parámetros input del programa mql5, estáticas y automáticas. En caso de necesidad cada variable es inicializada por el valor correspondiente. Después de ser usada, la variable se deinicializa, y la memoria ocupada por ésta se devuelve al sistema ejecutivo MQL5.

Inicialización y deinicialización de variables globales

Las variables globales se inicializan de una forma automática inmediatamente después de que se cargue el programa mql5 y antes de que se llame a cualquier función. Durante la inicialización se asignan valores iniciales a las variables de tipos simples y se llama al constructor (si existe) para los objetos. Los parámetros input siempre se declaran a nivel global, son inicializados por valores asignados por los usuarios en un diálogo durante el inicio del programa mql5.

A pesar de que las variables estáticas suelen declararse a nivel local, la memoria para estas variables se distribuye de antemano, y la inicialización se realiza justo después de que el programa se cargue, lo mismo que pasa con las variables globales.

El orden de inicialización corresponde al orden de declaración de una variable en el programa, y la deinicialización se realiza a la inversa antes de cargar el programa. Esta regla vale sólo para la variables que no han sido creadas por el operador new. Estas variables se crean y se inicializan automáticamente justo después de la carga, y se deinicializan directamente antes de la descarga del programa.

Inicialización y deinicialización de variables locales

Si una variable declarada a nivel local no es estática, entonces la distribución de memoria para esta variable se realiza de una forma automática. Las variables locales, igual como globales, se inicializan automáticamente en el momento cuando la ejecución del programa encuentra la declaración de una variable local. De esta manera, el orden de inicialización corresponde al orden de declaración.

Las variables locales se deinicializan al final del bloque del programa en el cual ha sido declaradas, y en el orden inverso a la declaración. Un bloque del programa es un operador compuesto que puede ser una parte del operador de elección switch, ciclo (for, while, do-while), cuerpo de función o parte del operador if-else.

Las variables locales se inicializan sólo en el momento cuando la ejecución del programa llega a la declaración de variable. Si durante el proceso de ejecución del programa el bloque en el que una variable está declarada, no ha sido ejecutado, esta variable no se inicializa.

Inicialización y deinicialización de objetos colocados dinámicamente

Un caso especial representan los punteros a objetos, puesto que la declaración de un puntero no implica la posterior inicialización del objeto correspondiente. Los objetos colocados dinámicamente se inicializan sólo en el momento cuando la muestra de clase es creada por el operador new. La inicialización del objeto supone la invocación del constructor de la clase correspondiente. Si en la clase

© 2000-2025, MetaQuotes Ltd.

no hay constructores correspondientes, entonces sus elementos que tienen el tipo simple, no serán inicializados automáticamente; los elementos de tipos string, array dinámico y objeto compuesto serán inicializados de una forma automática.

Los punteros pueden ser declarados a nivel local o global y pueden ser inicializados por el valor vacío NULL o por el valor del puntero del mismo tipo o del tipo derivado. Si el operador *new* ha sido invocado para un puntero declarado a nivel local, entonces el operador *delete* para este puntero tiene que ser ejecutado antes de salir de este nivel. En caso contrario, el puntero se perderá y el objeto no podrá ser eliminado de una forma explícita.

Todos los objetos creados por la expresión *puntero_objeto*=new *Nombre_Clase*, a continuación tienen que ser eliminados obligatoriamente por el operador delete(*puntero_objeto*). Si por alguna razón esta variable no ha sido eliminada por el operador delete después de que el programa complete su trabajo, aparecerá un mensaje de entrada en el apartado "Expertos". Es posible declarar varias variables, y asignarles a todas el puntero a un objeto.

Si el objeto creado dinámicamente tiene un constructor, este constructor será llamado en el momento de la ejecución del operador *new*. Si el objeto tiene un destructor, este destructor será llamado en el momento de la ejecución del operador *delete*.

De esta manera, los objetos colocados dinámicamente se crean sólo en el momento cuando son creados por el operador *new*, y se eliminan de una forma garantizada por el operador *delete* o automáticamente por el sistema ejecutora de MQL5 durante la descarga del programa. El orden de declaración de punteros de los objetos creados dinámicamente no influye en el orden de su inicialización. El programador controla completamente el orden de inicialización y deinicialización.

Particularidades de trabajo con la memoria dinámica

Durante el trabajo con los arrays dinámicos la memoria liberada se devuelve inmediatamente al sistema operativo.

Cuando se crea un objeto dinámico de la clase a través del operador new, primero la memoria se solicita desde el pool de la memoria de las clases con el que trabaja el asistente de memoria. Si en el pool no hay memoria suficiente, entonces ésta se solicita en el sistema operativo. Cuando un objeto dinámico se elimina a través del operador delete, la memoria que ocupaba este objeto se devuelve al pool de la memoria de las clases.

El asistente de la memoria devuelve la memoria al sistema inmediatamente después de la salida de las funciones-manejadores de eventos: OnInit(), OnDeinit(), OnStart(), OnTick(), OnCalculate(), OnTimer(), OnTrade(), OnTester(), OnTesterInit(), OnTesterPass(), OnTesterDeinit(), OnChartEvent(), OnBookEvent().

Características breves de variables

La información general sobre el orden de creación, eliminación, llamada a los constructores y destructores se muestra en la tabla de abajo.

© 2000-2025, MetaQuotes Ltd.

	Variable global automática	Variable local automática	Objeto creado dinámicamente
Inicialización	inmediatamente después de que se cargue el programa mql5	al llegar durante la ejecución a la línea de código, donde ésta está declarada	durante la ejecución del operador **new**
Orden de inicialización	en el orden de declaración	en el orden de declaración	no depende del orden de declaración
Deinicialización	antes de que se descargue el programa mql5	cuando la ejecución abandona el bloque de declaración	durante la ejecución del operador **delete** o antes de que se descargue el programa mql5
Orden de deinicialización	en el orden inverso a la inicialización	en el orden inverso a la inicialización	no depende del orden de inicialización
Llamada al constructor	al cargar el programa mql5	durante la inicialización	durante la ejecución del operador *new*
Llamada al destructor	al descargar el programa mql5	al salir del bloque donde la variable ha sido inicializada	durante la ejecución del operador *delete*
Aviso de errores	aviso en el apartado "Expertos" sobre el intento de eliminación del objeto creado automáticamente	aviso en el apartado "Expertos" sobre el intento de eliminación del objeto creado automáticamente	aviso en el apartado "Expertos" sobre los objetos creados dinámicamente pero con fallos durante la descarga del programa mql5

Véase también

Tipos de datos, Encapsulación y extensión de tipos, Inicialización de variables, Visibilidad y tiempo de vida de variables

© 2000-2025, MetaQuotes Ltd.

Preprocesador

El preprocesador es un subsistema especial del compilador MQL5 que se ocupa de la preparación preliminar del código fuente del programa directamente antes de su compilación.

El preprocesador permite mejorar la legibilidad del código fuente. Se puede conseguir la estructurización del código mediante la inclusión de ciertos archivos con los códigos fuentes de los programas mql5. Además de eso, la posibilidad de asignar los nombres mnemotécnicos a algunas constantes también contribuye a la mejora de legibilidad del código.

El preprocesador, asimismo, permite determinar los parámetros especiales de los programas mql5:

- Declarar constantes
- Establecer propiedades de programa
- Incluir archivos en texto de programa
- Importar funciones
- Usar compilación condicional

El compilador usa las directivas del preprocesador para procesar de manera preliminar el código fuente antes de proceder a su compilación. La directiva comienza siempre con el símbolo # (almohadilla), por este motivo, el compilador prohíbe utilizar este símbolo en los nombres de las variables, funciones, etcétera.

Cada directiva se describe con una entrada aparte y es válida hasta el salto de línea. No es posible usar varias directivas en una sola entrada. Si la entrada de la directiva es demasiado grande, podremos dividirla en varias líneas con la ayuda de la barra invertida '\', en este caso, la línea siguiente se considerará una continuación de la entrada de la directiva.

```
//+------------------------------------------------------------------+
//|  pseudooperador foreach                                          |
//+------------------------------------------------------------------+
#define ForEach(index, array) for (int index = 0,                     \
   max_##index=ArraySize((array));                                    \
   index<max_##index; index++)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string array[]={"12","23","34","45"};
//--- rastreamos la matriz con la ayuda de ForEach
   ForEach(i,array)
     {
      PrintFormat("%d: array[%d]=%s",i,i,array[i]);
     }
  }
//+------------------------------------------------------------------+
/* Resultado mostrado
   0: array[0]=12
   1: array[1]=23
```

© 2000-2025, MetaQuotes Ltd.

```
   2: array[2]=34
   3: array[3]=45
*/
```

Para el compilador, estas tres líneas de la directiva #define tendrán el aspecto de una línea larga. En este ejemplo también se usa el símbolo de almohadilla doble ##, llamado operador de fusión. Este se utiliza en los macros #define para unir los dos tokens de un macro en uno solo. El operador de fusión de tokens no puede ser el primero ni el último en la definición del macro.

© 2000-2025, MetaQuotes Ltd.

Declaración de constante (#define, #undef)

El compilador usa las directivas del preprocesador para procesar de manera preliminar el código fuente antes de proceder a su compilación. La directiva comienza siempre con el símbolo # (almohadilla), por este motivo, el compilador prohíbe utilizar este símbolo en los nombres de las variables, funciones, etcétera.

Cada directiva se describe con una entrada aparte y es válida hasta el salto de línea. No es posible usar varias directivas en una sola entrada. Si la entrada de la directiva es demasiado grande, podremos dividirla en varias líneas con la ayuda de la barra invertida '\', en este caso, la línea siguiente se considerará una continuación de la entrada de la directiva.

La directiva #define puede ser utilizada para la asignación de los nombres mnemotécnicos a las constantes. Hay dos formas:

```
#define identifier expression                    // forma libre de parámetros
#define identifier(par1,... par8) expression     // forma paramétrica
```

La directiva #define sustituye todas las siguientes entradas identifier en el texto original por **expression**. identifier se sustituye sólo en el caso si es un token suelto. identifier no se sustituye si es parte del un comentario, una cadena o parte de otro identificador más largo.

El identificador de constante se rige por las mismas reglas, las que funcionan para los nombres de las variables. El valor puede ser de cualquier tipo:

```
#define ABC               100
#define PI                3.14
#define COMPANY_NAME      "MetaQuotes Software Corp."
...
void ShowCopyright()
  {
   Print("Copyright  2001-2009, ",COMPANY_NAME);
   Print("https://www.metaquotes.net");
  }
```

expression puede estar compuesto por varios tokens, como por ejemplo, palabras claves, constantes, expresiones constantes y no constantes. **expression** se termina con el fin de la línea y no puede saltar a la siguiente.

Ejemplo:

```
#define TWO         2
#define THREE       3
#define INCOMPLETE TWO+THREE
#define COMPLETE   (TWO+THREE)
void OnStart()
  {
   Print("2 + 3*2 = ",INCOMPLETE*2);
   Print("(2 + 3)*2 = ",COMPLETE*2);
  }
/* Resultado
   2 + 3*2 = 8
```

© 2000-2025, MetaQuotes Ltd.

```
   (2 + 3)*2 = 10
*/
```

Forma paramétrica #define

En caso de forma paramétrica, todas las siguientes entradas encontradas del identifier serán reemplazadas por expression, tomando en consideración los parámetros actuales. Por ejemplo,

```
 // un ejemplo con dos parámetros a y b
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))

double c=MUL(A,B);
Print("c=",c);
/*
expresión double c=MUL(A,B);
es equivalente a double c=((2+3)*(5-1));
*/
// Resultado
// c=20
```

Asegúrese de encerrar los parámetros entre paréntesis a la hora de usarlos en expression, porque esto va a permitirle evitar los errores implícitos, difíciles de detectar. Si reescribimos el ejemplo sin utilizar los paréntesis, el resultado será totalmente diferente:

```
 // un ejemplo con dos parámetros a y b
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
expresión double c=MUL(A,B);
es equivalente a double c=2+3*5-1;
*/
// Resultado
// c=16
```

Cuando se utiliza la forma paramétrica, se admite el máximo de 8 parámetros.

```
 // forma paramétrica correcta
#define LOG(text)  Print(__FILE__,"(",__LINE__,") :",text)   // un parámetro - 'text'

 // forma paramétrica incorrecta
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9)   p1+p2+p3+p4 // más de 8 paráme
```

Directiva #undef

Directiva #undef sirve para cancelar la macro declarada anteriormente.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#define MACRO

void func1()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MACRO is not defined in ",__FUNCTION__);
#endif
  }

#undef MACRO

void func2()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MACRO is not defined in ",__FUNCTION__);
#endif
  }

void OnStart()
  {
   func1();
   func2();
  }

/* Resultado:
 MACRO is defined in func1
 MACRO is not defined in func2
*/
```

Véase también

Identificadores, Constantes de caracteres

© 2000-2025, MetaQuotes Ltd.

Propiedades de programas (#property)

Para cada programa mql5 se puede indicar unos parámetros específicos adicionales *#property*, los que ayudan al terminal de cliente prestar un servicio correcto a los programas sin necesidad de iniciarlos explícitamente. En primer lugar, esto se refiere a los ajustes de los indicadores externos. Las propiedades descritas en los archivos incluidos se ignoran por completo. Las propiedades tienen que ser especificadas en el archivo-mql5 principal.

```
#property identificador valor
```

El compilador apuntará los valores renovados en los ajustes del módulo ejecutado.

Constante	Tipo	Descripción
icon	string	ruta del archivo con la imagen que va a mostrarse como el icono del programa EX5. Las reglas para especificar la ruta son las mismas que para los recursos. La propiedad tiene que especificarse en el módulo principal con el código fuente MQL5. El archivo del icono debe ser del formato ICO.
link	string	Referencia a la web de la empresa productora
copyright	string	Nombre de la empresa productora
version	string	Versión del programa, no más de 31 caracteres
description	string	Breve descripción textual del programa mql5. Puede haber algunas description, cada una de las cuales describe una línea de texto. La longitud total de todas las description no puede superar más de 511 caracteres incluyendo saltos de líneas
stacksize	int	Indica el tamaño de la pila para el programa MQL5. La pila de un volumen grande es necesaria cuando se ejecutan las llamadas recursivas a la función. Cuando en el gráfico se inicia un script o un EA, se asigna la pila como mínimo de 8 Mb. Para En caso de los indicadores, la pila siempre tiene el tamaño fijo de 1 Mb. Cuando el programa se inicia en el Probador de Estrategias, se le asigna la pila de 16 Mb.
library		Biblioteca; no asigna ninguna función de inicio; funciones con modificador export se puede importar en otros programas mql5
indicator_applied_price	int	Especifica el valor por defecto para el campo "Apply to". Se puede especificar uno de los valores de enumeración ENUM_APPLIED_PRICE. Si la propiedad no está especificada, entonces por defecto se aplica el valor PRICE_CLOSE

© 2000-2025, MetaQuotes Ltd.

Constante	Tipo	Descripción
indicator_chart_window		Mostrar el indicador en la ventana del gráfico
indicator_separate_window		Mostrar el indicador en la ventana separada
indicator_height	int	El alto fijo de la subventana del indicador en píxeles (propiedad INDICATOR_HEIGHT)
indicator_buffers	int	Cantidad de búferes para el cálculo del indicador
indicator_plots	int	Cantidad de series gráficas en el indicador
indicator_minimum	double	Margen inferior de escala de la ventana separada del indicador
indicator_maximum	double	Margen superior de escala de la ventana separada del indicador
indicator_labelN	string	Pone una marca para la enésima serie gráfica mostrada en la ventana DataWindow. Para las series gráficas que necesitan varios búferes de indicador (DRAW_CANDLES, DRAW_FILLING y los demás), los nombres de las marcas vienen con separador ';'.
indicator_colorN	color	Color para mostrar la línea N, donde la N es el número de la serie gráfica; la enumeración se empieza desde 1
indicator_widthN	int	Grosor de línea en la serie gráfica, donde la N es el número de la serie gráfica; la enumeración se empieza desde 1
indicator_styleN	int	Estilo de línea la serie gráfica especificado con el valor de ENUM_LINE_STYLE. La N es el número de la serie gráfica, la enumeración se empieza desde 1
indicator_typeN	int	Tipo de construcción gráfica especificado con el valor de ENUM_DRAW_TYPE. La N es el número de la serie gráfica, la enumeración se empieza desde 1
indicator_levelN	double	Nivel horizontal N en la ventana separada del indicador
indicator_levelcolor	color	Color de niveles horizontales del indicador
indicator_levelwidth	int	Grosor de niveles horizontales del indicador
indicator_levelstyle	int	Estilo de niveles horizontales del indicador
script_show_confirm		Mostrar ventana de confirmación antes de iniciar un script
script_show_inputs		Mostrar ventana con las propiedades antes de iniciar un script y prohibir la muestra de la ventana de confirmación

© 2000-2025, MetaQuotes Ltd.

Constante	Tipo	Descripción
tester_indicator	string	Nombre del indicador personalizado en el formato "*nombre_de_indicador.ex5*". Los indicadores que requieren verificaciones se definen automáticamente desde la llamada a la función iCustom(), si el parámetro correspondiente es fijado por una cadena constante. Para los demás casos (uso de la función IndicatorCreate() o uso de una cadena no constante en el parámetro que asigna el nombre del indicador) se necesita esta propiedad
tester_file	string	Nombre del archivo incluyendo su extensión, que se pasa al comprobador para que éste lo procese, tiene que ir encerrado entre comillas dobles (como una cadena constante). Siempre hay que especificar los archivos de entrada, en caso de que sean necesarios
tester_library	string	Nombre de la biblioteca con su extensión encerrado entre comillas dobles. Una biblioteca puede tener tanto la extensión dll como la ex5. Las bibliotecas que necesitan verificación se determinan automáticamente. Sin embargo, si una de las bibliotecas se usa por un indicador personalizado, hace falta usar esta propiedad
tester_set	string	Nombre del archivo set con los valores y el salto de los parámetros de entrada. El archivo especificado se transmitirá al simulador antes de la prueba o la optimización. El nombre del archivo se debe indicar con la extensión correspondiente y comillas dobles, como una línea de constante. Si en el nombre del archivo set indicamos el nombre del experto y el número de la versión "<expert_name>_<number>.set", el archivo se añadirá automáticamente al menú de carga de las versiones de los parámetros con el número de versión <number>. Por ejemplo, el nombre "MACD Sample_4.set" indica un archivo set para el experto "MACD Sample.mq5" cuyo número de versión es 4. Para analizar el formato, recomendamos guardar manualmente los ajustes de simulación/optimización en el simulador de estrategias y después abrir el archivo set creado de esa forma.
tester_no_cache	string	El simulador de estrategias, al realizar la optimización, guarda todos los resultados de las

© 2000-2025, MetaQuotes Ltd.

Constante	Tipo	Descripción
		pasadas ejecutadas en la caché de optimización. En esta se guarda el resultado de la simulación para cada conjunto de parámetros de entrada. Esto permite tomar los resultados preparados al realizar de nuevo una optimización con los mismos parámetros, sin que haya necesidad de realizar nuevos cálculos ni perder tiempo. Pero, para algunas tareas – por ejemplo, con los cálculos matemáticos – podría ser necesario realizar los cálculos independientemente de que haya resultados preparados en la caché de optimización. En este caso, deberemos incluir en el archivo la propiedad *tester_no_cache*. En este caso, además, los resultados de la simulación se guardarán de todas formas en la caché, para que sea posible consultar en el simulador de estrategias todos los datos de las pasadas ejecutadas.
tester_everytick_calculate	string	En el simulador de estrategias, los indicadores se calculan solo al recurrir a los mismos, es decir, solo en el momento cuando se solicitan los valores de los búferes de indicador. Esto proporciona un aumento de velocidad considerable al realizar la simulación y la optimización, si no es necesario obtener los valores del indicador en cada tick. Durante la simulación, la indicación de la propiedad *tester_everytick_calculate* permite activar forzosamente el modo de cálculo del indicador en cada tick. Los indicadores en el simulador de estrategias también se calculan forzosamente en cada tick en los siguientes casos: • al realizar la simulación en el modo visual; • si en el indicador existen las funciones EventChartCustom, OnChartEvent, OnTimer; • si el indicador ha sido creado con un compilador cuyo número de build sea inferior a 1916. Esta propiedad se refiere solo al trabajo en el simulador de estrategias, en el terminal, los indicadores siempre se calculan en cada tick entrante.

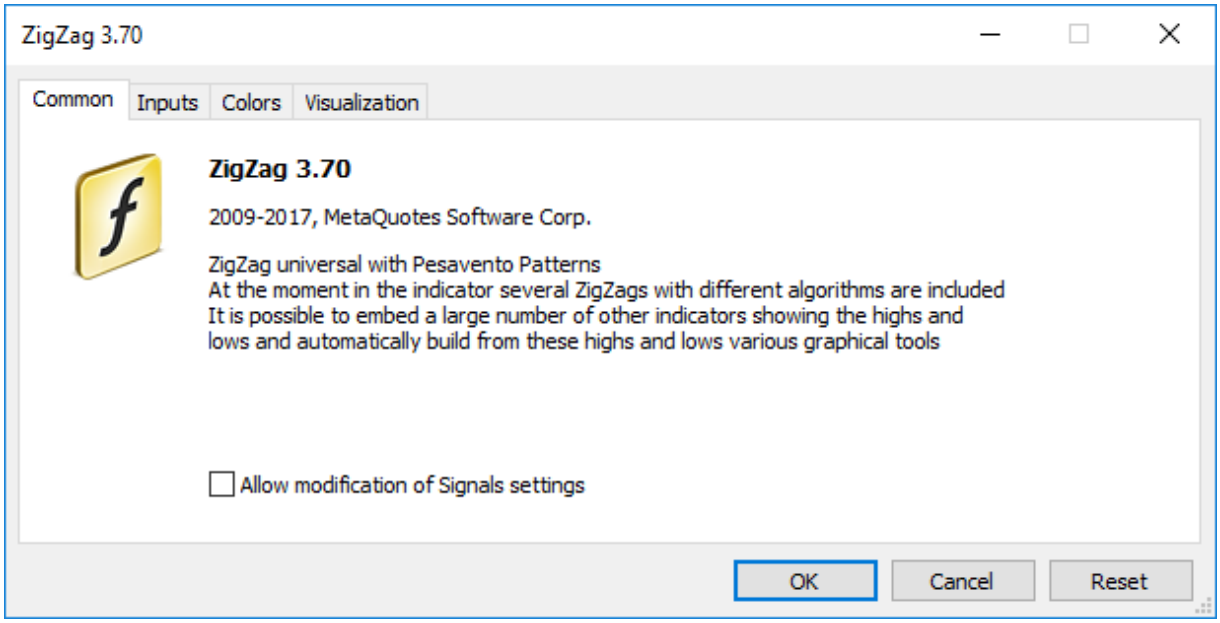
© 2000-2025, MetaQuotes Ltd.

Constante	Tipo	Descripción
optimization_chart_mode	string	Indica el tipo del gráfico y los nombres de los dos parámetros de entrada que se utilizarán para visualizar los resultados de la optimización. Por ejemplo, "3d, InpX, InpY" indica que se mostrará un gráfico 3D con ejes de coordenadas basados en los valores de los parámetros iterados InpX y InpY. De esta forma, con la ayuda de esta propiedad, será posible indicar directamente en el código los parámetros que se usarán para mostrar el gráfico de optimización y el tipo del propio gráfico. Posibles variantes: • "3d, input_parameter_name1, input_parameter_name2" - gráfico 3D de visualización con volumen; se puede rotar, acercar y alejar. Se construye según dos parámetros de entrada. • "2d, input_parameter_name1, input_parameter_name2" - gráfico plano en forma de cuadrícula; cada celda está coloreada en función del resultado. Se construye según dos parámetros de entrada. • "1d, input_parameter_name1, input_parameter_name2" - gráfico lineal en el que los resultados se clasifican según el parámetro indicado. Cada pasada se representa en forma de punto, y se construye según un parámetro de entrada. • "0d, input_parameter_name1, input_parameter_name2" - gráfico normal con resultados; estos se clasifican siguiendo el orden de llegada de los resultados de la pasada. Cada pasada se representa en forma de punto. No necesita que se indiquen parámetros, pero los parámetros establecidos se usarán al cambiar manualmente a otro tipo de gráfico. Es posible indicar el tipo del gráfico y no indicar uno o dos parámetros de entrada; en ese caso, el terminal elegirá por sí mismo los parámetros de entrada para mostrar el gráfico de optimización.

Ejemplo de descripción y número de una versión

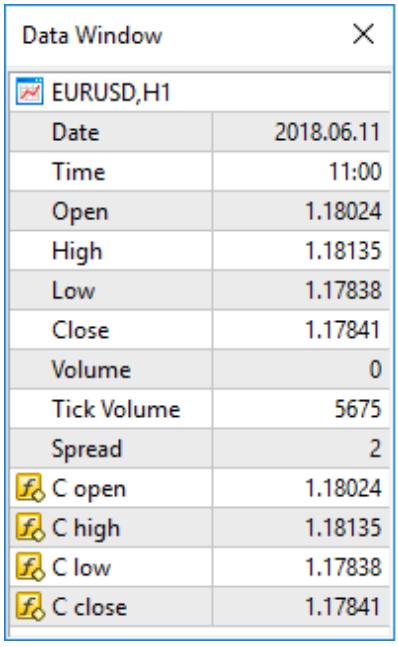
```
#property version     "3.70"      // versión actual de Asesor Experto
#property description "ZigZag universal con patrones de Pesavento"
#property description "Actualmente en el indicador están integrados varios ZigZag con
#property description "Existe posibilidad de integrar una cantidad importante de otros
#property description "mínimos, y construir automáticamente a base de éstos diferentes
```

© 2000-2025, MetaQuotes Ltd.



Ejemplo de especificar una marca separada para cada búfer de indicadores ("C open;C high;C low;C close")

```
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_plots   1
#property indicator_type1   DRAW_CANDLES
#property indicator_width1  3
#property indicator_label1  "C open;C high;C low;C close"
```





© 2000-2025, MetaQuotes Ltd.

Inclusión de archivos (#include)

La línea de comando *#include* puede encontrarse en cualquier parte del programa, pero normalmente las inclusiones suelen encontrarse al principio del archivo del código fuente. Formato de llamada:

```
#include <nombre_de_archivo>
#include "nombre_de_archivo"
```

Ejemplos:

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

El preprocesador sustituye la cadena *#include <nombre_de_archivo>* por el contenido del archivo WinUser32.mqh. Los paréntesis angulares significan que el archivo WinUser32.mqh va a ser cogido desde un directorio estándar (suele ser *directorio_de_terminal\MQL5\Include*). El directorio corriente no se mira.

Si el nombre del archivo se encierra entre las comillas dobles, entonces la búsqueda se efectua en el directorio corriente (donde se encuentra el archivo principal del código fuente). El directorio estándar no se mira.

Véase también

Biblioteca estándar, Importación de funciones

© 2000-2025, MetaQuotes Ltd.

Importación de funciones (#import)

Las funciones se importan desde los módulos compilados MQL5 (archivos *.ex5) y desde los módulos del sistema operativo (archivos *.dll). El nombre del módulo se indica en la directiva *#import*. Para que el compilador pueda hacer correctamente la llamada a la función importada y organizar el correcto traspaso de parámetros, hace falta la descripción completa de las funciones. La descripción de la función sigue directamente después de la directiva *#import "nombre de módulo"*. El nuevo comando *#import* (puede ir sin parámetros) cierra el bloque de descripción de las funciones importadas.

```
#import "nombre_de_archivo"
    func1 define;
    func2 define;
    ...
    funcN define;
#import
```

Las funciones importadas pueden tener cualquier nombre. Las funciones con nombres iguales pueden ser importadas a la vez desde diferentes módulos. Las funciones importadas pueden tener nombres que coincidan con los nombres de las funciones built-in. La operación de resolución de contexto determina cuál de las funciones tiene que ser invocada.

La orden de búsqueda del archivo especificado tras la palabra clave #import se describe en la sección Llamadas a las funciones importadas.

Dado que las funciones importadas se encuentran fuera del módulo compilado, el compilador no puede comprobar si los parámetros pasados son correctos. Así que para evitar los errores de ejecución, hay que describir con exactitud la composición y el orden de los parámetros pasados a las funciones importadas. Los parámetros pasados a funciones importadas (tanto de EX5, como de DLL-módulos) pueden tener valores por defecto.

En funciones importadas no se puede usar lo siguiente en calidad de parámetros:

- punteros (*);
- referencias a objetos que contienen arrays dinámicos y/o punteros.

Las clases, arrays de cadenas o objetos complejos que contienen las cadenas y/o arrays dinámicos de cualquier tipo no se puede pasar como parámetros a las funciones importadas de DLL.

Ejemplos:

```
#import "stdlib.ex5"
string ErrorDescription(int error_code);
int    RGB(int red_value,int green_value,int blue_value);
bool   CompareDoubles(double number1,double number2);
string DoubleToStrMorePrecision(double number,int precision);
string IntegerToHexString(int integer_number);
#import "ExpertSample.dll"
int    GetIntValue(int);
double GetDoubleValue(double);
string GetStringValue(string);
double GetArrayItemValue(double &arr[],int,int);
bool   SetArrayItemValue(double &arr[],int,int,double);
double GetRatesItemValue(double &rates[][6],int,int,int);
```

© 2000-2025, MetaQuotes Ltd.

```
#import
```

Para importar funciones durante la ejecución del programa mql5 se usa la aligación temprana. Eso significa que la biblioteca se carga durante el proceso de carga del programa ex5 que la utilice.

No se recomienda usar el nombre totalmente calificado del módulo cargable del tipo *Drive:\Directory\FileName.Ext*. Las bibliotecas MQL5 se cargan de la carpeta *terminal_dir\MQL5\Libraries*.

Si la función importada tiene diferentes variantes de llamada para las versiones de 32 y 64 bits de Windows, entonces será necesario importar ambas y llamar de forma explícita la variante necesaria de la función con la ayuda de la variable _IsX64.

Ejemplo:

```
#import "user32.dll"
//--- para el sistema de 32 bits
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
//--- para el sistema 64 bits
int    MessageBoxW(ulong hWnd,string lpText,string lpCaption,uint uType);
#import
//+------------------------------------------------------------------+
//|  MessageBox_32_64_bit usa la variante necesaria de MessageBoxW()   |
//+------------------------------------------------------------------+
int MessageBox_32_64_bit()
  {
   int res=-1;
   //--- si tenemos la versión de 64 bits de Windows
   if(_IsX64)
     {
      ulong hwnd=0;
      res=MessageBoxW(hwnd,"Ejemplo de llamada de la versión de 64 bits de MessageBoxW
     }
   else  // tenemos la versión de 32 bits de Windows
     {
      uint hwnd=0;
      res=MessageBoxW(hwnd,"Ejemplo de llamada de la versión de 32 bits de MessageBoxW
     }
   return (res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int ans=MessageBox_32_64_bit();
   PrintFormat("MessageBox_32_64_bit returned %d",ans);
  }
```

© 2000-2025, MetaQuotes Ltd.

Importando funciones de bibliotecas .NET

Para trabajar con las funciones de una biblioteca .NET, basta con importar la propia DLL sin indicar funciones concretas. El MetaEditor importará automáticamente todas las funciones con las que se pueda trabajar:

- Las estructuras sencillas (POD, plain old data), son estructuras que contienen solo tipos sencillos de datos.
- Las funciones estáticas públicas en cuyos parámetros se usan solo tipos sencillos y estructuras POD o sus matrices.

Para llamar las funciones de una biblioteca, solo tiene que importarlas:

```
#import "TestLib.dll"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int x=41;
   TestClass::Inc(x);
   Print(x);
  }
```

El código C# de la función Inc de la clase TestClass tiene el aspecto siguiente:

```
public class TestClass
  {
   public static void Inc(ref int x)
     {
      x++;
     }
  }
```

Como resultado de la ejecución, el script retornará el valor 42.

Véase también

Inclusión de archivos

© 2000-2025, MetaQuotes Ltd.

Compilación condicional (#ifdef, #ifndef, #else, #endif)

El compilador usa las directivas del preprocesador para procesar de manera preliminar el código fuente antes de proceder a su compilación. La directiva comienza siempre con el símbolo # (almohadilla), por este motivo, el compilador prohíbe utilizar este símbolo en los nombres de las variables, funciones, etcétera.

Cada directiva se describe con una entrada aparte y es válida hasta el salto de línea. No es posible usar varias directivas en una sola entrada. Si la entrada de la directiva es demasiado grande, podremos dividirla en varias líneas con la ayuda de la barra invertida '\', en este caso, la línea siguiente se considerará una continuación de la entrada de la directiva.

Las directivas de compilación condicional del preprocesador permiten compilar u omitir una parte del programa dependiendo del cumplimiento de una cierta condición.

Esta condición puede tomar una de las siguientes formas.

```
#ifdef identifier
   // el código ubicado aquí se compila si identifier ya ha sido definido para el prep
#endif
```

```
#ifndef identifier
   // el código ubicado aquí se compila si identifier en este momento no está definid
#endif
```

Después de cualquier directiva de compilación condicional puede ir cualquier número de líneas que posiblemente contengan la directiva #else y se terminan con #endif. Si la condición a comprobar es cierta, las líneas entre #else y #endif se ignoran. Si la condición a comprobar no se cumple, se ignoran todas las líneas entre la comprobación y la directiva #else (si ésta no existe, entre la comprobación y la directiva #endif).

Ejemplo:

```
#ifndef TestMode
   #define TestMode
#endif
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef TestMode
      Print("Test mode");
   #else
      Print("Normal mode");
   #endif
  }
```

En función del tipo del programa y el modo de compilación, las macros estándar se definen de la siguiente manera:

© 2000-2025, MetaQuotes Ltd.

La macro __MQL5__ se define cuando se compila *.mq5, la macro __MQL4__ se define cuando se compila *.mq4.
La macro _DEBUG se define cuando se compila en modo de depuración.
La macro _RELEASE se define cuando se compila en modo de realización.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef __MQL5__
      #ifdef _DEBUG
         Print("Hello from MQL5 compiler [DEBUG]");
      #else
        #ifdef _RELEASE
           Print("Hello from MQL5 compiler [RELEASE]");
        #endif
      #endif
   #else
      #ifdef __MQL4__
        #ifdef _DEBUG
           Print("Hello from MQL4 compiler [DEBUG]");
        #else
           #ifdef _RELEASE
              Print("Hello from MQL4 compiler [RELEASE]");
           #endif
        #endif
      #endif
   #endif
  }
```

© 2000-2025, MetaQuotes Ltd.

Programación orientada a objetos

Programación orientada a objetos (POO) es programación enfocada a los datos, siendo de notar que los datos y el comportamiento están vinculados indisolublemente entre si. Los datos juntamente con el comportamiento constituyen una clase, y los objetos son instancias de clases.

Los componentes de la programación orientada a objetos son los siguientes:

- Encapsulación y extensión de tipos
- Herencia
- Polimorfismo
- Sobrecarga
- Funciones virtuales

POO considera los cálculos como el modelado del comportamiento. Lo que se modela es un objeto representado por las abstracciones computacionales. Supongamos que queremos escribir el juego muy conocido a todos "Tetris", para eso tenemos que aprender a modelar la aparición de una figura aleatoria compuesta por cuatro cuadrados unidos juntos por sus bordes. Además, hace falta regular la velocidad de caída de esta figura, definir las operaciones de rotación y desplazamiento. En una pantalla los movimientos de figuras están limitados con los márgenes del área de juego, este requisito también debe ser modelado. Aparte de eso, las líneas completas deben desaparecer del vaso y hay que llevar la cuenta de puntos obtenidos durante el juego.

De esa manera, un juego tan fácil de comprender requiere la creación de varios modelos: modelo de pieza, modelo de vaso, modelo de movimiento dentro del vaso, etc. Todos estos modelos son unas abstracciones representadas por los cálculos en el ordenador. Para describir estos modelos se utiliza el concepto de *tipo de dato abstracto, TDA* (o tipo de dato compuesto). Hablando en rigor, el modelo de movimiento de una "figura" en el "vaso" no es un tipo de datos, sino un conjunto de operaciones con los datos de tipo "figura" que utilizan limitaciones de datos de tipo "vaso".

Los objetos son variables de clase. Programación orientada a objetos permite crear y usar TDA de una manera fácil. Programación orientada a objetos utiliza el mecanismo de herencia. La ventaja de herencia consiste en el hecho de que se permite la obtención de tipos derivados de los tipos de datos ya definidos por el usuario. Así que, para crear figuras en tetris, para empezar, es más cómodo crear la clase base Shape, a base de la cual se obtienen los tipos derivados de siete posibles figuras de tetris. El comportamiento de las figuras está determinado en la clase base, mientras que en las derivadas se concreta la realización del comportamiento de cada una de las figuras.

En POO los objetos se hacen responsables de su comportamiento. El programador de TDA debe incluir en él un código para describir cualquier comportamiento que normalmente se puede esperar de los objetos correspondientes. El hecho de que el mismo objeto se responsabilice de su comportamiento facilita considerablemente la tarea de programación para cada usuario de este objeto.

Si queremos dibujar una figura en la pantalla, hemos de saber dónde va a estar su centro y cómo dibujarla. Si una figura separada entiende perfectamente como dibujar a si misma, el programador, al usar esta figura, solo debe enviar al objeto el mensaje "dibujar".

El lenguaje MQL5 es parecido al C++, y también dispone del mecanismo de encapsulación para implementación de TDA. Encapsulación reune en sí, por una parte, los detalles internos de implementación de un tipo en particular, y por otra parte, las funciones accesibles desde fuera que

© 2000-2025, MetaQuotes Ltd.

puedan influir sobre los objetos de este tipo. Los detalles de implementación pueden ser inaccesibles para el programa que utilice este tipo.

Un conjunto de conceptos tiene relación con el concepto de POO, incluyendo los siguientes:

- Simulación de acciones del mundo real
- Disponibilidad de tipos de datos definidos por el usuario
- Ocultación de detalles de implementación
- Posibilidad de usar el código varias veces gracias a la herencia
- Interpretación de la llamada a la función durante la ejecución

Algunos de estos conceptos son bastante vagos, algunos son abstractos, otros son de carácter general.

© 2000-2025, MetaQuotes Ltd.

Encapsulación y extensión de tipos

POO es una manera equilibrada de enfocar la escritura del software. Los datos y el comportamiento están empaquetados juntos. Esta encapsulación crea tipos de datos definidos por el usuario, que amplían los tipos de datos del lenguaje y se interactuan. La extensión de tipos es una posibilidad de agregar al lenguaje los tipos de datos definidos por el usuario, los cuales también pueden ser usados de una manera fácil, igual que los tipos básicos.

Un tipo de dato abstracto, por ejemplo una cadena, es una descripción del comportamiento ideal, muy bien conocido. El usuario de la cadena sabe que las operaciones, tales como la concatenación o la impresión, tienen un cierto comportamiento. Las operaciones de concatenación e impresión se llaman métodos.

La implementación concreta de TDA puede tener ciertas restricciones; por ejemplo, las cadenas pueden ser limitadas en su longitud. Estas limitaciones afectan el comportamiento abierto para todos. Al mismo tiempo, los detalles internos o privados de implementación no afectan directamente el modo como el usuario ve el objeto. Por ejemplo, una cadena a menudo se implementa como un array; mientras que la dirección básica interna de los elementos de este array y su nombre no son tan significativos para el usuario.

La encapsulación es la capacidad de ocultar los detalles internos cuando se proporciona un interfaz abierto al tipo definido por el usuario. En MQL5, igual que en C++, para la provisión de encapsulación se utilizan las definiciones de clase y estructura (class y struct) en combinación con las palabras claves de acceso private (privado), protected (protegido) y public (público).

La palabra clave public indica que el acceso a los elementos que la siguen está abierto sin ningunas restricciones. Sin esta palabra clave los elementos de la clase están cerrados por defecto. Los elementos cerrados están disponibles sólo para las funciones miembro de su clase.

Los elementos de clase protegidos están disponibles para las funciones miembro no sólo de su clase, sino también de las clases heredadas. Los elementos de acceso abiertos están disponibles para cualquier función dentro de la zona de visibilidad de declaración de la clase. La protección permite ocultar una parte de implementación de clase, evitando de esa manera los cambios imprevistos de la estructura de datos. La restricción de acceso o ocultación de datos son particularidades de la programación orientada a objetos.

Habitualmente procuran proteger los elementos de clase y declararlos con el modificador protected. El establecimiento y la lectura de valores de estos elementos se realiza usando los métodos llamados método-set y método-get, los cuales se definen con el modificador de acceso public.

Ejemplo:

```
class CPerson
  {
protected:
   string            m_name;                    // nombre
public:
   void              SetName(string n){m_name=n;}// establece el nombre
   string            GetName(){return (m_name);} // devuelve el nombre
  };
```

© 2000-2025, MetaQuotes Ltd.

Este enfoque ofrece varias ventajas. Primero, por el nombre de la función se puede entender qué es lo que hace: establece o recibe el valor de un elemento de clase. Segundo, tal vez en el futuro tengamos necesidad de cambiar el tipo de variable m_name en la misma clase CPerson o en alguna de sus clases derivadas.

En este caso, bastará con cambiar la implementación de la función SetName() y GetName(), mientras que los objetos de la clase CPerson se podrá usar en el programa sin ningunas modificaciones en el código, porque el usuario ni siquiera va a enterarse de que el tipo de datos m_name se haya cambiado.

Ejemplo:

```
struct Name
  {
   string            first_name;                  // nombre
   string            last_name;                   // apellido
  };

class CPerson
  {
protected:
   Name              m_name;                      // nombre
public:
   void              SetName(string n);
   string            GetName(){return(m_name.first_name+" "+m_name.last_name);}
private:
   string            GetFirstName(string full_name);
   string            GetLastName(string full_name);
  };

void CPerson::SetName(string n)
  {
   m_name.first_name=GetFirstName(n);
   m_name.last_name=GetLastName(n);
  }

string CPerson::GetFirstName(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>0) StringSetCharacter(full_name,pos,0);
   return(full_name);
  }

string CPerson::GetLastName(string full_name)
  {
   string ret_string;
   int pos=StringFind(full_name," ");
   if(pos>0) ret_string=StringSubstr(full_name,pos+1);
   else      ret_string=full_name;
   return(ret_string);
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

Véase también

Tipos de datos

© 2000-2025, MetaQuotes Ltd.

Herencia

La particularidad de POO es la promoción de reuso del código utilizando los mecanismos de herencia. Una clase nueva se crea de una ya existente que se llama la clase base (superclase). La clase derivada (subclase) utiliza los elementos de la clase base. La clase derivada usa los elementos de la clase base, aunque también puede modificar y completarlos.

Muchos tipos son variaciones de los temas ya existentes. A menudo resulta muy agotador elaborar un código nuevo para cada uno de ellos. Además, un código nuevo supone errores nuevos. La clase derivada hereda la descripción de la clase base, siendo innecesario volver a crear y comprobar el código. Las relaciones de herencia se representan en una forma jerárquica.

La jerarquía es un método que permite copiar los elementos en toda su diversidad y complejidad. Ella introduce la clasificación de objetos. Por ejemplo, en la tabla periódica de los elementos hay gases. Ellos poseen propiedades inherentes a todos los elementos del sistema.

Los gases inertes es la siguiente subclase importante. La jerarquía consiste en que el gas inerte, como por ejemplo el argón, es un gas, y en su lugar el gas es un elemento del sistema. Este tipo de jerarquía permite explicar fácilmente el comportamiento de los gases inertes. Sabemos que sus átomos contienen protones y electrones, lo que es cierto para los demás elementos.

Sabemos que se encuentran en el estado gaseoso con la temperatura interior, igual que todos los gases. Sabemos que ningún gas de la subclase de gases inertes entra en la reacción química común con ninguno de los elemento, y esta es la propiedad de todos los gases inertes.

Vamos a ver la jerarquía en el ejemplo de las figuras geométricas. Para describir la variedad de figuras simples (círculo, triángulo, rectángulo, cuadrado, etc.) es mejor crear una clase base (TDA), que resulta ser el antecesor de todas las clases derivadas.

Vamos a crear la clase base CShape que contiene sólo los miembros más comunes que describen la figura. Estos miembros describen las propiedades características de cualquier figura, es decir, tipo de figura y las coordenadas del punto principal de enlace.

Ejemplo:

```
//--- Clase base Figura
class CShape
  {
protected:
   int       m_type;                    // tipo de figura
   int       m_xpos;                    // X - coordenada del punto de enlace
   int       m_ypos;                    // Y - coordenada del punto de enlace
public:
             CShape(){m_type=0; m_xpos=0; m_ypos=0;} // constructor
   void      SetXPos(int x){m_xpos=x;} // establezcamos X
   void      SetYPos(int y){m_ypos=y;} // establezcamos Y
  };
```

Luego vamos a crear nuevas clases derivadas de la clase base en las cuales vamos a añadir campos necesarios que especifican cada clase en particular. Para la figura Circle(círculo) hace falta añadir un miembro que contiene el valor del radio. La figura Quadrate (cuadrado) se caracteriza por el valor del

© 2000-2025, MetaQuotes Ltd.

lado del cuadrado. Por lo tanto, las clases derivadas, heredadas de la clase base CShape, serán declaradas como sigue:

```
//--- clase derivada Círculo
class CCircle : public CShape        // después de dos puntos definimos la clase base
  {                                    // de la que se hace la herencia
private:
   int             m_radius;           // radio del círculo

public:
                   CCircle(){m_type=1;}// constructor, el tipo es igual a 1
  };
```

Para el cuadrado la declaración de la clase es similar:

```
//--- clase derivada Cuadrado
class CSquare : public CShape        // después de dos puntos definimos la clase base
  {                                    // de la que se hace la herencia
private:
   int            m_square_side;       // lado del cuadrado

public:
                   CSquare(){m_type=2;} // constructor, el tipo es igual a 2
  };
```

Cabe señalar que durante la creación de un objeto primero se llama al constructor de la clase base, y luego al constructor de la clase derivada. Cuando se destruye un objeto, primero se llama al destructor de la clase derivada, y luego al destructor de la clase base.

De esa manera, al declarar en la clase base los miembros más generales, podemos añadir miembros adicionales en las clases derivadas, los que precisan una clase en concreto. La herencia permite crear potentes bibliotecas de código que pueden ser utilizadas varias veces y en repetidas ocasiones.

La sintaxis de crear una clase derivada de una ya existente es como sigue:

```
class nombre_de_clase :
          (public | protected | private) opt  nombre_de_clase_base
  {
    declaración de miembros
  };
```

Uno de los aspectos de una clase derivada es la visibilidad (abertura) de sus miembros-herederos. Las palabras claves public, protected y private se utilizan para indicar el nivel de acceso de los elementos de la clase base para la clase derivada. El uso de la palabra clave private seguida de dos puntos en el encabezado de la clase derivada significa que los miembros protegidos y abiertos (protected y public) de la clase base CShape deberían ser heredados como elementos protegidos y abiertos de la clase derivada CCircle.

Los miembros privados de la clase base no están disponibles para la clase derivada. La herencia pública también significa que las clases derivadas (CCircle y CSquare) son CShape. Es decir, el cuadrado (CSquare) es la figura (CShape) pero la figura no tiene que ser obligatoriamen6te un cuadrado.

© 2000-2025, MetaQuotes Ltd.

La clase derivada es una modificación de la clase base; hereda los miembros protegidos y públicos de la clase base. Los constructores y destructores de la clase base no pueden ser heredados. A menudo a la clase derivada se añaden nuevos miembros como complemento a los miembros de la clase base.

La clase derivada puede incluir la implementación de las funciones-miembro diferente de la clase base. Eso no tiene nada que ver con la sobrecarga cuando el significado del nombre de la función puede ser distinto para diferentes signaturas.

Con la herencia protegida los miembros públicos y protegidos de la clase base se hacen miembros protegidos de la clase derivada. Con la herencia privada los miembros públicos y protegidos de la clase base se hacen miembros privados de la clase derivada.

Con la herencia protegida y privada la relación de que "un objeto de clase derivada es un objeto de clase base" no es cierta. La herencia protegida y privada se encuentran muy raras veces, y hay que usar cada una de ellas con mucho cuidado.

Hay que entender que el tipo de herencia (public, protected o private) de ninguna manera influye en los modos de **acceso a los miembros de las clases base en la jerarquía de la herencia desde una clase derivada (un heredero)**. Sea como sea el tipo de herencia, desde las clases derivadas estarán disponibles sólo los miembros de la clase base declarados con los especificadores de acceso public y protected. Veremos lo arriba mencionado en un ejemplo:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| Una clase de ejemplo con varios tipos de acceso                  |
//+------------------------------------------------------------------+
class CBaseClass
  {
private:              //--- el miembro privado no está disponible desde las clases deri
   int                m_member;
protected:            //--- el modo protegido está disponible desde la clase base y sus
   int                Member(){return(m_member);}
public:               // el constructor de la clase está disponible para todos
                      CBaseClass(){m_member=5;return;};
private:              // el modo cerrado para asignar valores al miembro m_member
   void               Member(int value) { m_member=value;};

  };
//+------------------------------------------------------------------+
//|  clase derivada con errores                                      |
//+------------------------------------------------------------------+
class CDerivaed: public CBaseClass // se puede omitir la especificación de la herencia
  {
public:
   void Func() // definimos en la clase derivada una función con las llamadas a los mi
     {
      //--- intento de modificación de un miembro privado de la clase base
      m_member=0;         // error, el miembro privado de la clase base no está disponi
```

© 2000-2025, MetaQuotes Ltd.

```
      Member(0);          // error, el método privado de la clase base no está disponib
      //--- lectura del miembro de la clase base
      Print(m_member);    // error, el miembro privado de la clase base no está disponi
      Print(Member());    // no hay error, el método público de la clase base está disp
     }
  };
```

En el ejemplo citado, la clase CBaseClass tiene sólo un método público, es el constructor. Los constructores se llaman automáticamente cuando se crea un objeto de la clase. Por eso no hay ninguna manera de llamar al miembro privado m_member, ni tampoco al método protegido Member() desde fuera. Pero en caso de la herencia pública (public), el método Member() de la clase base estará disponible desde las clases derivadas.

En caso de la herencia protegida (protected), todos los miembros de la clase base con el acceso público y protegido se hacen protegidos. Esto significa que si los miembros-datos y métodos públicos de la clase base estaban disponibles desde fuera, entonces ahora en caso de la herencia protegida ellos estarán disponibles sólo desde las clases del heredero y de sus posteriores clases derivadas.

```
//+------------------------------------------------------------------+
//| Una clase de ejemplo con varios tipos de acceso                  |
//+------------------------------------------------------------------+
class CBaseMathClass
  {
private:             //--- el miembro privado no está disponible desde las clases deri
   double            m_Pi;
public:              //--- obtención y establecimiento del parámetro para m_Pi
   void              SetPI(double v){m_Pi=v;return;};
   double            GetPI(){return m_Pi;};
public:              // el constructor de la clase está disponible para todos
                     CBaseMathClass() {SetPI(3.14);  PrintFormat("%s",__FUNCTION__);};
  };
//+------------------------------------------------------------------+
//| Una clase derivada en la que ya no se puede modificar m_Pi       |
//+------------------------------------------------------------------+
class CProtectedChildClass: protected CBaseMathClass // herencia protegida
  {
private:
   double            m_radius;
public:              //--- métodos públicos en la clase derivada
   void              SetRadius(double r){m_radius=r; return;};
   double            GetCircleLength(){return GetPI()*m_radius;};
  };
//+------------------------------------------------------------------+
//|  Función del inicio del script                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- durante la creación de una clase derivada, el constructor de la clase base se li
   CProtectedChildClass pt;
//--- especificamos el radio
```

© 2000-2025, MetaQuotes Ltd.

```
   pt.SetRadius(10);
   PrintFormat("Length=%G",pt.GetCircleLength());
//--- si comentamos la cadena de abajo, obtendremos el error en la fase de compilació
// pt.SetPI(3);

//--- ahora declararemos una variable de la clase base e intentaremos establecer la c
   CBaseMathClass bc;
   bc.SetPI(10);
//--- veremos lo que ha salido
   PrintFormat("bc.GetPI()=%G",bc.GetPI());
  }
```

En este ejemplo se muestra que los métodos SetPI() y GetPi() en la clase base CBaseMathClass son públicos y están disponibles para la llamada desde cualquier parte del programa. Pero al mismo tiempo, para su derivada CProtectedChildClass las llamadas a estos métodos se puede realizar sólo desde los métodos de la misma clase CProtectedChildClass o sus clases derivadas.

En caso de la herencia privada (private), todos los miembros de la clase base con el acceso public y protected se hacen privados, y en caso de posteriores herencias resulta imposible llamarlos.

En MQL5 no hay herencia múltiple.

Véase también

Estructuras y clases

© 2000-2025, MetaQuotes Ltd.

Polimorfismo

Polimorfismo es una posibilidad para objetos de diferentes clases vinculados por medio de la herencia reaccionar de varias maneras a la hora de dirigirse a la misma función-elemento. Eso permite crear unos mecanismos universales que describen el comportamiento no sólo de la clase base, sino el de las clases descendentes.

Vamos a seguir desarrollando la clase base CShape donde definimos la función miembro GetArea() que sirve para calcular la superficie de una figura. En todas las clases descendientes de la clase base vamos a redefinir esta función de acuerdo con las normas de cálculo de la superficie de una figura en concreto.

Para el cuadrado (clase CSquare) la superficie se calcula por los lados, para el círculo (clase CCircle) la superficie se calcula por el radio y etc. Podemos crear un array para almacenar los objetos del tipo CShape en el que podremos almacenar tanto el objeto de la clase base, como el de todos sus descendientes. En el futuro podremos llamar a la misma función para cualquier elemento de este array.

Ejemplo:

```
//--- Clase base
class CShape
  {
protected:
   int            m_type;                    // tipo de figura
   int            m_xpos;                    // X - coordenada del punto de enlace
   int            m_ypos;                    // Y - coordenada del punto de enlace
public:
   void           CShape(){m_type=0;};       // constructor, el tipo es igual a cero
   int            GetType(){return(m_type);};// devuelve el tipo de figura
virtual
   double         GetArea(){return (0); }    // devuelve la superficie de figura
  };
```

Ahora todas las clases derivadas tienen la función miembro getArea() que devuelve el valor cero. La implementación de esta función va a ser distinta en cada un de los descendientes.

```
//--- clase derivada Círculo
class CCircle : public CShape               // después de dos puntos definimos la clas
  {                                         // de la que se hace la herencia
private:
   double         m_radius;                 // radio del círculo

public:
   void           CCircle(){m_type=1;};     // constructor, el tipo es igual a 1
   void           SetRadius(double r){m_radius=r;};
   virtual double GetArea(){return (3.14*m_radius*m_radius);}// superficie del círculo
  };
```

Para el cuadrado la declaración de la clase es similar:

```
//--- clase derivada Cuadrado
```

© 2000-2025, MetaQuotes Ltd.

```
class CSquare : public CShape              // después de dos puntos definimos la clase
  {                                        // de la que se hace la herencia
private:
   double          m_square_side;          // lado del cuadrado

public:
   void            CSquare(){m_type=2;};   // constructor, el tipo es igual a 2
   void            SetSide(double s){m_square_side=s;};
   virtual double GetArea(){return (m_square_side*m_square_side);}//superficie del cua
  };
```

Como para el cálculo de la superficie del círculo y cuadrado se necesitan los valores correspondientes de los miembros m_radius y m_square_side, entonces en la declaración de la clase correspondiente hemos añadido las funciones SetRadius y SetSide().

Se supone que en nuestro programa se utilizan los objetos de deferentes tipos (CCircle y CSquare) pero heredados de un tipo base CShape. El polimorfismo no permite crear un array de objetos del tipo base CShape pero durante la declaración de este array los objetos aún no se conoces y su tipo no está definido.

La decisión sobre el objeto de qué tipo va a contenerse en cada elemento del array va a tomarse directamente en el proceso de ejecución del programa. Esto supone la creación dinámica de objetos de las clases correspondientes, y por consecuencia, la necesidad de usar los punteros a objetos en vez de los objetos en sí.

Para la creación dinámica de los objetos se utiliza el operador new. Cada objeto tiene que eliminarse de manera individual y explícita con el operador delete. Por eso declararemos un array de punteros del tipo CShape, y crearemos para cada elemento suyo un objeto del tipo necesario (**new Nombre _de_la_clase)**, tal como se muestra en el ejemplo del script:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declararemos un array de punteros del tipo base
   CShape *shapes[5];   // array de punteros a objetos CShape

//--- aquí llenamos el array con objetos derivados
//--- declararemos un puntero al objeto del tipo CCircle
   CCircle *circle=new CCircle();
//--- establecemos las propiedades del objeto según el puntero circle
   circle.SetRadius(2.5);
//--- colocaremos el valor del puntero en shapes[0]
   shapes[0]=circle;

//--- crearemos otro objeto CCircle más y escribiremos su puntero en shapes[1]
   circle=new CCircle();
   shapes[1]=circle;
   circle.SetRadius(5);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- aquí "hemos olvidado" establecer aposta el valor para shapes[2]
//circle=new CCircle();
//circle.SetRadius(10);
//shapes[2]=circle;

//--- estableceremos el valor NULL para el elemento que no se utiliza
   shapes[2]=NULL;

//--- crearemos el objeto CSquare y escribiremos su puntero en shapes[3]
   CSquare *square=new CSquare();
   square.SetSide(5);
   shapes[3]=square;

//--- crearemos el objeto CSquare y escribiremos su puntero en shapes[4]
   square=new CSquare();
   square.SetSide(10);
   shapes[4]=square;

//--- tenemos un array de punteros, obtendremos su tamaño
   int total=ArraySize(shapes);
//--- pasaremos en un ciclo por todos los punteros en el array
   for(int i=0; i<5;i++)
     {
      //--- si el puntero en el índice especificado es válido
      if(CheckPointer(shapes[i])!=POINTER_INVALID)
        {
         //--- mostraremos en el log el tipo y la superficie del figura
         PrintFormat("El objeto del tipo %d tiene la superficie %G",
               shapes[i].GetType(),
               shapes[i].GetArea());
        }
      //--- si el puntero tiene el tipo POINTER_INVALID
      else
        {
         //--- avisaremos sobre el error
         PrintFormat("¡El objeto shapes[%d] no ha sido inicializado! Su puntero es %s"
                     i,EnumToString(CheckPointer(shapes[i])));
        }
     }

//--- tenemos que eliminar personalmente todos los objetos dinámicos creados
   for(int i=0;i<total;i++)
     {
      //--- se puede eliminar sólo los objetos cuyo puntero tiene el tipo POINTER_DYNA
      if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
        {
         //--- avisaremos sobre la eliminación
         PrintFormat("Eliminamos shapes[%d]",i);
         //--- eliminaremos los objetos según su puntero
```

© 2000-2025, MetaQuotes Ltd.

```
            delete shapes[i];
         }
      }
   }
```

Preste la atención a que cuando se elimina un objeto por el operador delete, hay que comprobar el tipo de su puntero. A través de delete se puede eliminar sólo los objetos que tienen los punteros POINTER_DYNAMIC. Para los punteros de otros tipos se mostrará el error.

Pero a parte de la redefinición de la función durante la herencia, el polimorfismo incluye también la implementación de la misma función con diferentes conjuntos de parámetros dentro de los límites de una clase. Eso significa que la clase puede tener varias funciones con el mismo nombre pero de diferentes tipos y/o conjunto de parámetros. En este caso el polimorfismo se implementa a través de la sobrecarga de funciones.

Véase también

Biblioteca estándar

© 2000-2025, MetaQuotes Ltd.

Sobrecarga

Dentro de los límites de una clase se puede definir dos o más métodos que conjuntamente usan el mismo nombre pero tienen diferente cantidad de parámetros. Cuando eso ocurre, los métodos se llaman *sobrecargados*, y del proceso se dice como de la *sobrecarga de método*. La sobrecarga de método es uno de los modos de realizar el polimorfismo. La sobrecarga de método en las clases se efectua según las mismas reglas que la sobrecarga de función.

Si no hay correspondencia exacta para la función llamada, el compilador busca la función conveniente en tres niveles de una manera consecutiva:

1. búsqueda entre los métodos de la clase;
2. búsqueda entre los métodos de las clases base, del ascendiente más próximo hasta el primero sucesivamente;
3. búsqueda entre las demás funciones.

Si no se ha encontrado la correspondencia exacta en ninguno de los niveles, pero hay unas funciones convenientes en los niveles diferentes, entonces se utiliza la función en el nivel menor. No puede haber más de una función conveniente dentro de los límites de un nivel.

En MQL5 no hay sobrecarga de operadores.

Véase también

Sobrecarga de funciones

© 2000-2025, MetaQuotes Ltd.

Funciones virtuales

La palabra clave virtual es un especificador de la función que proporciona el mecanismo para la elección dinámica durante la fase de ejecución de una función miembro conveniente entre las funciones de la clase base y derivada. Las estructuras no pueden tener funciones virtuales. Puede usarse para el cambio de las declaraciones sólo de las funciones miembro.

La función virtual, igual que una corriente, debe tener un cuerpo ejecutable. Durante la llamada su semántica es la misma que la de las demás funciones.

Una función virtual puede ser sustituida en una clase derivada. La elección de qué tipo de definición de la función llamar para una función virtual, se hace de una forma dinámica (durante la fase de ejecución). Un caso típico es cuando la clase base contiene una función virtual, y las clases derivadas tienen sus versiones de esta función.

El puntero a la clase base puede indicar al objeto de la clase base o al objeto de la clase derivada. La elección de la función miembro va a ser realizada en la fase de ejecución y va a depender del tipo del objeto y no del tipo del puntero. Si no hay miembro del tipo derivado, por defecto se utiliza la función virtual de la clase base.

Los destructores siempre son virtuales, independientemente de que si están declarados con la palabra clave virtual o no.

Vamos a ver el uso de funciones virtuales en el ejemplo del programa MT5_Tetris.mq5. La clase base CTetrisShape con la función virtual Draw (dibujar) está definida en el archivo incluido MT5_TetrisShape.mqh.

```
//+------------------------------------------------------------------+
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
```

© 2000-2025, MetaQuotes Ltd.

```
   void              Rotate()                    { m_prev_turn=m_turn; if(++m_turn>3) r
   virtual void      Draw()                      { return;                 }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

Luego, para cada clase derivada, esta función se implementa de acuerdo con las particularidades de la clase descendiente. Por ejemplo, la primera figura CTetrisShape1 tiene su propia implementación de la función Draw():

```
class CTetrisShape1 : public CTetrisShape
  {
public:
   //--- dibujo de figura
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      if(m_turn==0 || m_turn==2)
        {
         //--- vara horizontal
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
           }
        }
      else
        {
         //--- vara vertical
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+i*SHAPE_SIZE);
           }
        }
     }
  }
```

La figura cuadrado está descrita por la clase CTetrisShape6 y tiene su propia implementación del método Draw():

```
class CTetrisShape6 : public CTetrisShape
  {
public:
   //--- dibujo de figura
```

© 2000-2025, MetaQuotes Ltd.

```
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      for(i=0; i<2; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
        }
      for(i=2; i<4; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+(i-2)*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+SHAPE_SIZE);
        }
     }
  };
```

Dependiendo de qué clase ha sido creado el objeto, se llama a la función virtual de una u otra clase derivada.

```
void CTetrisField::NewShape()
  {
//--- creamos una de 7 posibles figuras de una manera aleatoria
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- dibujamos
   m_shape.Draw();
//---
  }
```

Modificador override

El modificador override indica que la función declarada deberá redefinir obligatoriamente el método de la clase padre. El uso de este modificador permite evitar errores en la redifinición, tales como el cambio casual de la signatura del método. Por ejemplo, en la clase básica se define el método func, que toma como argumento una variable del tipo int:

© 2000-2025, MetaQuotes Ltd.

```
class CFoo
  {
   void virtual func(int x) const { }
  };
```

A continuación, el método se redefine en la clase heredada:

```
class CBar : public CFoo
  {
   void func(short x) { }
  };
```

Pero por error, el tipo de argumento cambia de int a short. De hecho, en este caso ya tiene lugar no la redefinición, sino la sobrecarga del método. Actuando de acuerdo con el algoritmo de definición de la función sobrecargada, en ciertas situaciones el compilador puede elegir el método definido en la clase básica, en lugar del método redefinido.

Para evitar errores semejantes, al método redefinido se le debe añadir claramente el modificador override.

```
class CBar : public CFoo
  {
   void func(short x) override { }
  };
```

Si durante la redefinición se cambia la signatura del método, el compilador no podrá encontrar en la clase padre un método con la misma signatura y dará un error de compilación:

```
'CBar::func' method is declared with 'override' specifier but does not override any ba
```

Modificador final

El modificador final actúa al revés, prohibiendo la redefinición del método en las clases heredadas. Si la implementación de un método es autosuficiente y ha finalizado por completo, declárela con el modificador final, para que haya garantías de que no será modificada en lo sucesivo.

```
class CFoo
  {
   void virtual func(int x) final { }
  };

class CBar : public CFoo
  {
   void func(int) { }
  };
```

Al intentar redefinir un método con el modificador final, como se muestra en el ejemplo de más arriba, el compilador dará error:

```
'CFoo::func' method declared as 'final' cannot be overridden by 'CBar::func'
see declaration of 'CFoo::func'
```

Véase también

Biblioteca estándar

© 2000-2025, MetaQuotes Ltd.

Miembros estáticos de una clase/estructura

Miembros estáticos

Los miembros de una clase pueden ser declarados con el uso del modificador de la clase de memoria static. Estos miembros de datos se comparten por todos los ejemplares de esta clase y se guardan en un sitio. Los miembros de datos no estáticos se crean para cada variable-objeto de la clase.

La imposibilidad de declarar los miembros estáticos de una clase conduciría a la necesidad de declarar estos datos a nivel global del programa. Esto rompería las relaciones entre los datos y su clase, además de no concordarse con el paradigma básica de la POO - la unión de datos y métodos para su procesamiento dentro de una clase. El miembro estático permite existir a los datos de la clase que no son específicos para un ejemplar particular, en el campo de alcance de la clase.

Puesto que un miembro estático de la clase no depende de un ejemplar concreto, la referencia a él es como sigue:

```
class_name::variable
```

donde *class_name* es el nombre de la clase, y *variable* significa el nombre del miembro de la clase.

Como ve, para acceder al miembro estático de la clase, se utiliza el operador de resolución de contexto ::. Cuando accedemos a un miembro estático dentro de los métodos de la clase, el operador de contexto es opcional.

El miembro estático de la clase debe ser inicializado explícitamente con un valor necesario. Para eso hay que declarar e inicializarlo a nivel global. El orden de la inicialización de los miembros estáticos va a corresponder al orden de su declaración a nivel global en el código fuente.

Por ejemplo, tenemos una clase *CParser* que se utiliza para el análisis sintáctico de textos, y necesitamos obtener el número total de palabras y símbolos procesados. Sólo hay que declarar los miembros necesarios de la clase como estáticos e inicializarlos a nivel global. Entonces, todos los ejemplares de la clase van a utilizar durante su trabajo los contadores comunes de palabras y símbolos.

```
//+------------------------------------------------------------------+
//|  Clase "Analizador de textos"                                    |
//+------------------------------------------------------------------+
class CParser
  {
public:
   static int        s_words;
   static int        s_symbols;
   //--- constructor y destructor
                     CParser(void);
                    ~CParser(void){};
  };
...
//--- inicialización de los miembros estáticos de la clase Parser a nivel global
int CParser::s_words=0;
int CParser::s_symbols=0;
```

Un miembro estático de la clase puede ser declarado con la palabra clave *const*. Estas constantes estáticas tienen que ser inicializadas a nivel global con la palabra clave *const:*

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Clase "Pila" para almacenar los datos procesados                |
//+------------------------------------------------------------------+
class CStack
  {
public:
                     CStack(void);
                    ~CStack(void){};
...
private:
   static const int  s_max_length; // capacidad máxima de la pila
  };

//--- inicialización de la constante estática de la clase CStack
const int CStack::s_max_length=1000;
```

Puntero this

La palabra clave this denota un puntero declarado implícitamente a sí mismo - a un ejemplar concreto de la clase en contexto del cual se ejecuta el método. Se puede utilizarlo sólo en los métodos no estáticos de la clase. El puntero this es un miembro implícito no estático de cualquier clase.

En las funciones estáticas se puede acceder sólo a los miembros/métodos estáticos de la clase.

Métodos estáticos

En MQL5 se permite utilizar las funciones-miembros del tipo static. El modificador *static* debe ir antes del tipo de la función devuelto en la declaración dentro de la clase.

```
class CStack
  {
public:
   //--- constructor y destructor
                     CStack(void){};
                    ~CStack(void){};
   //--- capacidad máxima de la pila
   static int        Capacity();
private:
   int               m_length;     // número de elementos en la pila
   static const int  s_max_length; // capacidad máxima de la pila
  };
//+------------------------------------------------------------------+
//| Devuelve el número máximo de elementos para el almacenamiento en la pila|
//+------------------------------------------------------------------+
int CStack::Capacity(void)
  {
   return(s_max_length);
  }
//--- inicialización de la constante estática de la clase CStack
const int CStack::s_max_length=1000;
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos la variable del tipo CStack
   CStack stack;
//--- llamamos al método estático del objeto
   Print("CStack.s_max_length=",stack.Capacity());
//--- también se puede llamar de esta manera, ya que el método es estático y no requie
   Print("CStack.s_max_length=",CStack::Capacity());
  }
```

El método con modificador const se llama constante y no puede modificar los miembros implícitos de su clase. La declaración de las funciones constantes de la clase y los parámetros constantes se llama *control de constancia* (const-correctness). Gracias a este control se puede estar seguro de que el compilador va a controlar la constancia de valores de los objetos y mostrará error en la fase de compilación en caso de violación.

El modificador const se pone después de la lista de argumentos dentro de la declaración de la clase. La definición fuera de la clase también debe incluir el modificador *const*:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Clase "Rectángulo"                                               |
//+------------------------------------------------------------------+
class CRectangle
  {
private:
   double            m_width;      // ancho
   double            m_height;     // alto
public:
   //--- constructores y destructor
                     CRectangle(void):m_width(0),m_height(0){};
                     CRectangle(const double w,const double h):m_width(w),m_height(h)
                    ~CRectangle(void){};
   //--- cálculo de la superficie
   double            Square(void) const;
   static double     Square(const double w,const double h);// { return(w*h); }
  };
//+------------------------------------------------------------------+
//|  Devuelve la superficie del objeto "Rectángulo"                  |
//+------------------------------------------------------------------+
double CRectangle::Square(void) const
  {
   return(Square(m_width,m_height));
  }
//+------------------------------------------------------------------+
//|  Devuelve el resultado de multiplicación de dos variables
//+------------------------------------------------------------------+
static double CRectangle::Square(const double w,const double h)
  {
   return(w*h);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- creamos un rectángulo rect con los lados 5 y 6
   CRectangle rect(5,6);
//--- buscaremos la superficie del rectángulo utilizando el método constante
   PrintFormat("rect.Square()=%.2f",rect.Square());
//--- buscaremos el resultado de los números utilizando el método estático de la clase
   PrintFormat("CRectangle::Square(2.0,1.5)=%f",CRectangle::Square(2.0,1.5));
  }
```

Como argumento adicional a favor del uso del control de constancia sirve el hecho de que el compilador en este caso genera una optimización especial. Por ejemplo, coloca el objeto constante en la memoria sólo para la lectura.

Una función estática no puede ser determinada con el modificador const, puesto que este modificador garantiza la constancia de los miembros del ejemplar cuando se llama a esta función. Pero, como ya hemos dicho antes, una función estática por definición no puede acceder a los miembros no estáticos de la clase.

Véase también

Variables estáticas, Variables, Enlaces. Modificador & y la palabra clave this

© 2000-2025, MetaQuotes Ltd.

Plantillas de funciones

Las funciones sobrecargadas suelen utilizarse para la ejecución de operaciones semejantes con diferentes tipos de datos. La función ArraySize() es un sencillo ejemplo en el lenguaje MQL5. Ella devuelve el tamaño del array de cualquier tipo. En realidad, esta función del sistema es una función sobrecargada, y toda la implementación de esta sobrecarga está ocultada de los desarrolladores de programas MQL5:

```
int  ArraySize(
   void&  array[]      // array a comprobar
   );
```

Es decir, en realidad para cada invocación de esta función el compilador del lenguaje MQL5 inserta una implementación necesaria. Por ejemplo, así es como se puede hacer para los arrays del tipo entero:

```
int  ArraySize(
   int&  array[]      // array con los elementos del tipo int
   );
```

Y para el array del tipo MqlRates que se usa para el trabajo con las cotizaciones en el formato de datos históricos, la función ArraySize() puede representarse de la siguiente manera:

```
int  ArraySize(
   MqlRates&  array[] // array llenado con los valores del tipo MqlRates
   );
```

De esta manera, resulta muy cómodo utilizar la misma función para trabajar con diferentes tipos. Pero hace falta realizar todo el trabajo preliminar de manera individual, es decir: sobrecargar la función necesaria para todos los tipos de datos con los que tendrá que trabajar correctamente.

Hay una solución más conveniente. Si las operaciones idénticas deben realizarse para cada uno de los tipos de datos, la solución más compacta y cómoda sería el uso de las plantillas de funciones. En este caso, al programador le hace falta escribir solamente una descripción de la plantilla de la función. Cuando la plantilla se describe de esta manera, será suficiente especificar algún parámetro formal en vez de un determinado tipo de datos con los que debe trabajar la función. El compilador va a generar automáticamente diferentes funciones para el procesamiento correspondiente de cada tipo basándose en los tipos de los argumentos utilizados durante la invocación de la función.

La definición de la plantilla de una función se empieza con la palabra clave template seguida de la lista de los parámetros formales encerrados entre los corchetes angulares. Antes de cada parámetro formal se pone la palabra clave typename. Los tipos formales de los parámetros son los tipos built-in o los tipos definidos por el usuario. Se utilizan:

- para especificar los tipos de argumentos de la función,
- para especificar los tipos del valor devuelto de la función,
- para declarar variables dentro del cuerpo de descripción de la función.

El número de parámetros de una plantilla no podrá exceder de ocho. Cada parámetro formal de la definición de la plantilla tiene que aparecer en la lista de los parámetros de la función por lo menos una vez. Cada uno de los nombres del parámetro formal tiene que ser único.

© 2000-2025, MetaQuotes Ltd.

Aquí tenemos un ejemplo de una función para la búsqueda del valor máximo en el array de cualquier tipo numérico (números enteros y reales):

```
template<typename T>
T ArrayMax(T &arr[])
  {
   uint size=ArraySize(arr);
   if(size==0) return(0);

   T max=arr[0];
   for(uint n=1;n<size;n++)
      if(max<arr[n]) max=arr[n];
//---
   return(max);
  }
```

Esta plantilla define la función que busca el valor máximo en el array pasado y devuelve este valor como resultado. Vamos a recordar que la función ArrayMaximum() incorporada en MQL5 devuelve sólo el índice del valor máximo encontrado que será utilizado a continuación para obtener el mismo valor en cuestión. Por ejemplo:

```
//--- creamos un array
   double array[];
   int size=50;
   ArrayResize(array,size);
//---  lo llenamos con valores aleatorios
   for(int i=0;i<size;i++)
     {
      array[i]=MathRand();
     }

//--- buscamos la posición del elemento máximo en el array
   int max_position=ArrayMaximum(array);
//--- ahora obtenemos el mismo valor máximo en el array
   double max=array[max_position];
//--- visualizamos el valor encontrado
   Print("Max value = ",max);
```

De esta manera, hemos necesitado dos pasos para obtener el valor máximo del array. A través de la plantilla de la función ArrayMax() podemos obtener el resultado del tipo necesario con sólo pasar el array del tipo correspondiente a esta función. Es decir, en vez de las últimas dos líneas

```
//--- buscamos la posición del elemento máximo en el array
   int max_position=ArrayMaximum(array);
//--- ahora obtenemos el mismo valor máximo en el array
   double max=array[max_position];
```

ahora podemos usar una sola línea que va a devolver de una sola vez el resultado del mismo tipo que tiene el array pasado:

```
//--- buscamos el valor máximo
   double max=ArrayMax(array);
```

En este caso el tipo del resultado devuelto por la función ArrayMax() va a corresponder automáticamente al tipo del array.

© 2000-2025, MetaQuotes Ltd.

Es necesario usar la palabra clave typename para obtener el tipo del argumento en forma de una cadena, con el fin de crear modos universales de trabajo con diferentes tipos de datos. Vamos a demostrarlo con el ejemplo de una función que devuelve el tipo de datos en forma de una cadena:

```
#include <Trade\Trade.mqh>
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CTrade trade;
   double d_value=M_PI;
   int i_value=INT_MAX;
   Print("d_value: type=",GetTypeName(d_value), ",   value=", d_value);
   Print("i_value: type=",GetTypeName(i_value), ",   value=", i_value);
   Print("trade: type=",GetTypeName(trade));
//---
  }
//+------------------------------------------------------------------+
//| El tipo se devuelve como una línea                               |
//+------------------------------------------------------------------+
template<typename T>
string GetTypeName(const T &t)
  {
//--- devolvemos el tipo en forma de una línea
   return(typename(T));
//---
  }
```

Las plantillas de las funciones también se puede utilizar para los métodos de clase, por ejemplo:

```
class CFile
  {
   ...
public:
   ...
   template<typename T>
   uint WriteStruct(T &data);
  };

template<typename T>
uint CFile::WriteStruct(T &data)
  {
   ...
   return(FileWriteStruct(m_handle,data));
  }
```

Las plantillas de las funciones no se puede declarar con las palabras clave export, virtual y #import.

Sobrecarga de las funciones de plantilla

En algunos casos, podría ser necesaria la sobrecarga de la función de plantilla. Por ejemplo, tenemos una función de plantilla que registra en el primer parámetro el valor del segundo parámetro con la ayuda de la conversión explícita de tipos. En el lenguaje MQL5 está prohibida la conversión del tipo

© 2000-2025, MetaQuotes Ltd.

string al tipo bool, pero podemos hacerla por nosotros mismos, para ello, creamos la sobrecarga de la función de plantilla. Por ejemplo:

```
//+------------------------------------------------------------------+
//| Función de plantilla                                             |
//+------------------------------------------------------------------+
template<typename T1,typename T2>
string Assign(T1 &var1,T2 var2)
  {
   var1=(T1)var2;
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| Sobrecarga especial para el caso bool+string                     |
//+------------------------------------------------------------------+
string Assign(bool &var1,string var2)
  {
   var1=(StringCompare(var2,"true",false) || StringToInteger(var2)!=0);
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int i;
   bool b;
   Print(Assign(i,"test"));
   Print(Assign(b,"test"));
  }
```

Como resultado de la ejecución de este código, veremos que para la pareja int+string se ha usado la función de plantilla Assign(), y en la segunda llamada bool+string ya se ha usado la función sobrecargada.

```
string Assign<int,string>(int&,string)
string Assign(bool&,string)
```

Vea también

Sobrecarga

© 2000-2025, MetaQuotes Ltd.

Ventajas de las plantillas

Las plantillas de funciones se usan en aquellos casos en los que es necesario realizar operaciones iguales con datos de diferente tipo, por ejemplo, la búsqueda del elemento máximo en una matriz. La primera ventaja en el uso de las plantillas consiste en que el programador no tiene necesidad de escribir una sobrecarga aparte para cada tipo. Es decir, en lugar de declarar un conjunto de sobrecargas para cada tipo

```
double ArrayMax(double array[])
  {
   ...
  }
int ArrayMax(int array[])
  {
   ...
  }
uint ArrayMax(uint array[])
  {
   ...
  }
long ArrayMax(long array[])
  {
   ...
  }
datetime ArrayMax(datetime array[])
  {
   ...
  }
```

basta con escribir una función de plantilla

```
template<typename T>
T ArrayMax(T array[])
  {
   if(ArraySize()==0)
      return(0);
   uint max_index=ArrayMaximum(array);
   return(array[max_index]);
  }
```

y después usarla en su código:

```
double high[];
datetime time[];
....
double max_high=ArrayMax(high);
datetime lasttime=ArrayMax(time);
```

Aquí, el parámetro formal *T*, que establece el tipo de datos utilizado, se sustituye al compilar por el tipo utilizado realmente, es decir, el compilador genera automáticamente una función aparte para

© 2000-2025, MetaQuotes Ltd.

cada tipo double, datetime, etcétera. Exactamente de la misma forma se pueden crear en el lenguaje MQL5 las plantillas de clases, usando todas las ventajas de este enfoque.

Plantillas de clases

La plantilla de la clase se declara con la ayuda de la palabra clave template, tras la cual van las cuñas <>, dentro de los cuales se enumera la lista de parámetros formales con la palabra clave typename. Esta entrada indicará al compilador que ante él se encuentra una clase generalizada con el parámetro *T*, que establece el tipo real de la variable al implementar la clase. Por ejemplo, una clase-vector creada para guardar una matriz con elementos del tipo *T*:

```
#define TOSTR(x) #x+" "   // macro para mostrar el nombre del objeto
//+------------------------------------------------------------------+
//| Clase-vector del objeto para guardar los elementos del tipo T
//+------------------------------------------------------------------+
template <typename T>
class TArray
  {
protected:
   T                 m_array[];
public:
   //--- el constructor crea por defecto una matriz de 10 elementos
   void TArray(void){ArrayResize(m_array,10);}
   //--- constructor para crear un vector con un número de matriz establecido
   void TArray(int size){ArrayResize(m_array,size);}
   //--- retorna el tipo y la cantidad de datos que se guardan en el objeto del tipo T
   string Type(void){return(typename(m_array[0])+":"+(string)ArraySize(m_array));};
  };
```

A continuación, en el programa creamos de diferentes maneras tres objetos *TArray* para trabajar con diferentes tipos

```
void OnStart()
  {
   TArray<double> double_array;   // el vector tiene por defecto un tamaño 10
   TArray<int> int_array(15);     // el vector tiene un tamaño 15
   TArray<string> *string_array;  // puntero al vector TArray<string>
//--- creamos un objeto dinámico
   string_array=new TArray<string>(20);
//--- mostramos en el Diario el nombre del objeto, el tipo de datos y el tamaño del ve
   PrintFormat("%s (%s)",TOSTR(double_array),double_array.Type());
   PrintFormat("%s (%s)",TOSTR(int_array),int_array.Type());
   PrintFormat("%s (%s)",TOSTR(string_array),string_array.Type());
//--- eliminamos el objeto dinámico antes de finalizar el programa
   delete(string_array);
  }
```

Resultado de la ejecución del script:

```
  double_array  (double:10)
  int_array  (int:15)
```

© 2000-2025, MetaQuotes Ltd.

```
  string_array  (string:20)
```

Como resultado, se han creado 3 vectores con diferentes tipos de datos: double, int y string.

Las plantillas son adecuadas para desarrollar contenedores, es decir, objetos concebidos para encapsular objetos de cualquier tipo. Los objetos de los contenedores son colecciones que ya contienen objetos de un tipo determinado. Normalmente, en el contenedor se incorpora la implementación del trabajo con los datos que están guardados en el mismo.

Por ejemplo, se puede crear una plantilla de clase que no permita recurrir a un elemento fuera de la matriz, evitando de esta forma el error crítico "out of range".

```
//+------------------------------------------------------------------+
//| Clase para recurrir de forma segura a un elemento de la matriz   |
//+------------------------------------------------------------------+
template<typename T>
class TSafeArray
  {
protected:
   T                 m_array[];
public:
   //--- constructor por defecto
   void              TSafeArray(void){}
   //--- constructor para crear la matriz del tamaño establecido
   void              TSafeArray(int size){ArrayResize(m_array,size);}
   //--- tamaño de la matriz
   int               Size(void){return(ArraySize(m_array));}
   //--- cambio de tamaño de la matriz
   int               Resize(int size,int reserve){return(ArrayResize(m_array,size,rese
   //--- liberación de la matriz
   void              Erase(void){ZeroMemory(m_array);}
   //--- operador de acceso al elemento de la matriz según el índice
   T                 operator[](int index);
   //--- operador de atribución para obtener todos los elementos de la matriz a la vez
   void              operator=(const T  &array[]); // matriz del tipo T
  };
//+------------------------------------------------------------------+
//| Operación de obtención de un elemento según el índice            |
//+------------------------------------------------------------------+
template<typename T>
T TSafeArray::operator[](int index)
  {
   static T invalid_value;
//---
   int max=ArraySize(m_array)-1;
   if(index<0 || index>=ArraySize(m_array))
     {
      PrintFormat("%s index %d is not in range (0-%d)!",__FUNCTION__,index,max);
      return(invalid_value);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(m_array[index]);
  }
//+------------------------------------------------------------------+
//| Operación de atribución para la matriz                           |
//+------------------------------------------------------------------+
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   int size=ArraySize(array);
   ArrayResize(m_array,size);
//--- el tipo T debe dar soporte al operador de copiado
   for(int i=0;i<size;i++)
      m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int copied,size=15;
   MqlRates rates[];
//--- copiamos la matriz de cotizaciones
   if((copied=CopyRates(_Symbol,_Period,0,size,rates))!=size)
     {
      PrintFormat("CopyRates(%s,%s,0,%d) ha retornado el código de error %d",
      _Symbol,EnumToString(_Period),size,GetLastError());
      return;
     }
//--- creamos un contenedor e introducimos en él la matriz de valores MqlRates
   TSafeArray<MqlRates> safe_rates;
   safe_rates=rates;
   //--- índice dentro de los límites de la matriz
   int index=3;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
   //--- índice fuera de los límites de la matriz
   index=size;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
  }
```

Preste atención a que al describir los métodos fuera de los límites de la declaración de la clase, también es necesario usar la declaración de la plantilla:

```
template<typename T>
T TSafeArray::operator[](int index)
  {
   ...
  }
```

© 2000-2025, MetaQuotes Ltd.

```
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   ...
  }
```

Las plantillas de las clases y funciones permiten indicar varios parámetros formales separados con una coma, por ejemplo, la colección Map para guardar las parejas "clave - valor":

```
template<typename Key, template Value>
class TMap
  {
   ...
  }
```

Vea también

Plantillas de funciones, Sobrecarga

© 2000-2025, MetaQuotes Ltd.

Las clases abstractas y las funciones virtuales puras

Las clases abstractas están diseñadas para crear entidades genéricas, sobre cuya base se supone que se crearán clases derivadas más concretas. Una clase abstracta, es una clase que sólo puede ser utilizada como clase básica para alguna otra clase, por eso no se puede crear un objeto de tipo de clase abstracta.

La clase que contenga aunque sea solo una función virtual pura, es abstracta. Por eso las clases derivadas de una clase abstracta deben implementar todas sus funciones virtuales puras, de lo contrario, serán clases abstractas.

Una función virtual se declara como "pura" con la ayuda de la sintaxis de un especificador puro. Veremos como ejemplo la clase CAnimal, que se crea solo para proporcionar las funciones generales, los propios objetos del tipo CAnimal tienen un carácter demasiado general para la aplicación práctica. De esta forma, la clase CAnimal es un buen candidato a clase abstracta:

```
class CAnimal
  {
public:
                     CAnimal();      // constructor
   virtual void      Sound() = 0;    // función virtual pura
private:
   double            m_legs_count;   // número de patas del animal
  };
```

Aquí la función Sound() es virtual pura, por eso se la declara con el especificador de función virtual pura PURE (=0).

Las funciones virtuales puras son solo aquellas funciones virtuales para las que se indica el especificador puro PURE, y precisamente: (=NULL) o (=0). Ejemplo de declaración y uso de una clase abstracta:

```
class CAnimal
  {
public:
   virtual void      Sound()=NULL;   // PURE method, debe ser redefinido en la clase
  };
//--- derivada de la clase abstracta
class CCat : public CAnimal
 {
public:
  virtual void       Sound() { Print("Myau"); } // PURE es redefinido, la clase CCat
 };

//--- ejemplos de uso incorrecto
new CAnimal;          // error 'CAnimal' - el compilador retorna el error "cannot insta
CAnimal some_animal; // error 'CAnimal' - el compilador retorna el error "cannot insta

//--- ejemplos de uso correcto
new CCat;  // no hay error, la clase CCat no es abstracta
CCat cat;  // no hay error, la clase CCat no es abstracta
```

© 2000-2025, MetaQuotes Ltd.

Limitaciones de uso de las clases abstractas

Si el constructor de una clase abstracta invoca una función virtual pura (directa o indirectamente) el resultado es indefinido.

```
//+------------------------------------------------------------------+
//| Clase básica abstracta                                           |
//+------------------------------------------------------------------+
class CAnimal
  {
public:
   //--- función virtual pura
   virtual void      Sound(void)=NULL;
   //--- función
   void              CallSound(void) { Sound(); }
   //--- constructor
   CAnimal()
    {
     //--- invocación directa del método virtual
     Sound();
     //--- invocación indirecta (a través de una tercera función)
     CallSound();
     //--- un constructor y/o destructor siempre invoca sus propias funciones,
     //--- a pesar del carácter virtual y de la redefinición de la función invocada en
     //--- si la función invocada es virtual pura, entonces
     //--- la invocación provocará el error de ejecución crítico: "pure virtual functi
    }
  };
```

Sin embargo, los constructores y destructores de las clases abstractas pueden invocar otras funciones miembro.

© 2000-2025, MetaQuotes Ltd.

Espacios de nombres

Un espacio de nombres es una zona especialmente declarada, dentro de la cual se determinan distintos identificadores: variables funciones, clases, etcétera. Se establece con la ayuda de la palabra clave namespace:

```
namespace nombre_espacios {
  // lista de definiciones de funciones, clases y variables
}
```

El uso de namespace permite dividir el espacio global de nombres en varios espacios. Todos los identificadores dentro del espacio de nombres están disponibles unos para otros sin . Para acceder a los miembros del espacio de nombres, se usa el operador :: (operación de resolución de ámbito).

```
namespace ProjectData
{
class DataManager
  {
public:
   void              LoadData() {}
  };
void Func(DataManager& manager) {}
}
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- trabajando con el espacio de nombres ProjectData
   ProjectData::DataManager mgr;
   mgr.LoadData();
   ProjectData::Func(mgr);
  }
```

Los espacios de nombres se usan para organizar el código en forma de grupos lógicos y para evitar conflictos de nombres que pueden aparecer cuando se usan varias bibliotecas en el programa. En tales casos, podemos declarar cada biblioteca en su propio espacio de nombres, para después recurrir explícitamente a las funciones y clases necesarias de cada biblioteca.

El espacio de nombres puede declararse en varios bloques en un archivo o en varios de ellos. El compilador combina todas las partes durante el procesamiento preliminar, así que el espacio de nombres obtenido como resultado contendrá todos los miembros declarados en todas las partes. Supongamos que hemos implementado la clase A en el archivo de inclusión Sample.mqh:

```
//+------------------------------------------------------------------+
//|                                                   Sample.mqh |
//+------------------------------------------------------------------+
class A
  {
public:
                     A() {Print(__FUNCTION__);}
```

© 2000-2025, MetaQuotes Ltd.

```
  };
```

Queremos usar esta clase en nuestro proyecto, pero ya disponemos de una clase A. Para tener la posibilidad de usar ambas clases y evitar un conflicto de identificadores, bastará con envolver el archivo de inclusión en un espacio de nombres:

```
//--- declaramos la primera clase A
class A
  {
public:
                     A() {Print(__FUNCTION__);}
  };
//--- envolvemos la clae A del archivo "Sample.mqh" en el espacio de nombres "Library"
namespace Library
{
#include "Sample.mqh"
}
//--- añadimos otra clase más en el espacio de nombres "Library"
namespace Library
{
class B
  {
public:
                     B() {Print(__FUNCTION__);}
  };
}
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- usamos la clase A del espacio de nombres global
   A a1;
//--- usamos las clases A y B del espacio de nombres "Library"
   Library::A a2;
   Library::B b;
  }
//+------------------------------------------------------------------+

/*
Resultado:
   A::A
   Library::A::A
   Library::B::B
*/
```

Los espacios de nombres pueden ser anidados. Un espacio de nombres anidado tiene acceso ilimitado a los miembros de su espacio padre, pero los miembros del espacio padre no tienen acceso ilimitado al espacio de nombres anidado.

© 2000-2025, MetaQuotes Ltd.

```
namespace General
{
int Func();

namespace Details
{
 int Counter;
 int Refresh()  {return Func(); }
}

int GetBars()   {return(iBars(Symbol(), Period()));};
int Size(int i) {return Details::Counter;}
}
```

Espacio global de nombres

Si el identificador no ha sido explícitamente declarado en el espacio de nombres, se considerará que entra de forma implícita en el espacio global de nombres. Para indicar explícitamente un identificador global, use un operador de resolución de ámbito sin nombre. Esto permitirá distinguir este identificador de cualquier otro elemento con el mismo nombre que se encuentre en otro espacio de nombres. Por ejemplo, al importar una función:

```
#import "lib.dll"
int Func();
#import
//+------------------------------------------------------------------+
//|  Alguna función                                                  |
//+------------------------------------------------------------------+
int Func()
  {
   return(0);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//+--- llamando la función importada
   Print(lib::Func());
//+--- llamando nuestra función
   Print(::Func());
  }
```

En este caso, todas las funciones importadas desde la DLL han sido incluidas en el espacio de nombres homónimo. Esto ha permitido al compilador determinar de forma unívoca qué función debemos llamar.

Véase también

© 2000-2025, MetaQuotes Ltd.

Variables globales, Variables locales, Visibilidad y tiempo de vida de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

Constantes, enumeraciones y estruturas

Para facilitar la escritura del programa y para hacer la percepción de los textos fuentes de programas más cómoda, en el lenguaje MQL5 están previstas las constantes estándares predefinidas y las enumeraciones. Además, para almacenar la información se utilizan las estructuras auxiliares.

Las constantes estándares son similares a las macro sustituciones y tienen el tipo int.

Las constantes están agrupadas según su destinación:

- Constantes de gráficos se utilizan durante el trabajo con los gráficos de precios: apertura, navegación, fijación de parámetros;
- Constantes de objetos están destinadas para procesar los objetos gráficos que se puede crear y mostrar en los gráficos;
- Constantes de indicadores sirven para trabajar con indicadores estándares y de usuario ;
- Estado de ambiente describe las propiedades de un programa mql5, facilita la información sobre el terminal de cliente, instrumento financiero y cuenta corriente de trading;
- Constantes comerciales permiten precisar diversa información durante el proceso de comercio;
- Constantes nombradas son constantes del lenguaje MQL5;
- Estructuras de datos describen las formas utilizadas de almacenamiento de datos;
- Códigos de errores y advertencias describen mensajes del compilador y respuestas del servidor comercial a las peticiones comerciales;
- Constantes de entrada/salida están destinadas para trabajar con las funciones de archivo y para mostrar mensajes en la pantalla a través de la función MessageBox().

© 2000-2025, MetaQuotes Ltd.

Constantes de gráficos

Las constantes que describen diferentes propiedades de los gráficos están divididas en siguientes grupos:

- Tipos de eventos – eventos que ocurren a la hora de trabajar con los gráficos;
- Períodos de gráficos – períodos built-in estándares;
- Propiedades de gráficos – identificadores que se utilizan como parámetros de las funciones para trabajar con gráficos;
- Constantes de posicionamiento - valor del parámetro de la función ChartNavigate();
- Visualización de gráficos - configuración de la apariencia de gráfico.

© 2000-2025, MetaQuotes Ltd.

Tipos de eventos de gráfico

Existen 11 tipos de eventos que pueden ser procesados utilizando la función predefinida OnChartEvent(). Para los eventos de usuario están previstos 65535 identificadores en el rango de CHARTEVENT_CUSTOM a CHARTEVENT_CUSTOM_LAST inclusive. Para generar un evento de usuario hace falta usar la función EventChartCustom().

ENUM_CHART_EVENT

Identificador	Descripción
CHARTEVENT_KEYDOWN	Teclazos
CHARTEVENT_MOUSE_MOVE	Desplazamiento del ratón y pulsación de los botones del ratón (si para el gráfico está establecida la propiedad CHART_EVENT_MOUSE_MOVE=true)
CHARTEVENT_MOUSE_WHEEL	Pulsar o deslizar la rueda del ratón (si para el gráfico se ha establecido la propiedad CHART_EVENT_MOUSE_WHEEL=true)
CHARTEVENT_OBJECT_CREATE	Creación de un objeto gráfico (si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_CREATE=true)
CHARTEVENT_OBJECT_CHANGE	Evento de cambio de propiedades de un objeto gráfico a través del diálogo de propiedades
CHARTEVENT_OBJECT_DELETE	Eliminación de un objeto gráfico (si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_DELETE=true)
CHARTEVENT_CLICK	Clic en un gráfico
CHARTEVENT_OBJECT_CLICK	Clic en un objeto gráfico
CHARTEVENT_OBJECT_DRAG	Arrastrar un objeto gráfico
CHARTEVENT_OBJECT_ENDEDIT	Fin de edición del texto en el objeto gráfico Edit
CHARTEVENT_CHART_CHANGE	Modificación de dimensiones del gráfico o de sus propiedades a través del diálogo de propiedades
CHARTEVENT_CUSTOM	Número inicial de un evento del rango de eventos de usuario
CHARTEVENT_CUSTOM_LAST	Número final de un evento del rango de eventos de usuario

Para cada tipo del evento, los parámetros de entrada de la función OnChartEvent() tienen determinados valores que son necesarios para procesar este evento. En la tabla de abajo están especificados los eventos y valores que se pasan a través de los parámetros.

© 2000-2025, MetaQuotes Ltd.

Evento	Valor del parámetro id	Valor del parámetro lparam	Valor del parámetro dparam	Valor del parámetro sparam
Evento del teclado	CHARTEVENT_KEYDOWN	código de la tecla pulsada	Número de pulsaciones de la tecla generadas mientras ésta se mantenía en estado pulsado	Valor literal de la máscara de bits que describe el estatus de las teclas del teclado
Eventos del ratón (si para el gráfico está establecida la propiedad CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	coordinada X	coordinada Y	Valor literal de la máscara de bits que describe el estatus de los botones del ratón
Evento de rueda del ratón (si para el gráfico se ha establecido el evento CHART_EVENT_MOUSE_WHEEL=true)	CHARTEVENT_MOUSE_WHEEL	Banderas de estados de las teclas y los botones del ratón, coordenadas X e Y del ratón. Se da una descripción en el ejemplo de abajo	Valor Delta del deslizamiento de la rueda del ratón	—
Evento de creación de un objeto gráfico (si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_CREATE=true)	CHARTEVENT_OBJECT_CREATE	—	—	Nombre del objeto gráfico creado
Evento del cambio de propiedades de un objeto a través del diálogo de propiedades	CHARTEVENT_OBJECT_CHANGE	—	—	Nombre del objeto gráfico modificado
Evento de eliminación de un objeto gráfico	CHARTEVENT_OBJECT_DELETE	—	—	Nombre del objeto gráfico eliminado

© 2000-2025, MetaQuotes Ltd.

Evento	Valor del parámetro id	Valor del parámetro lparam	Valor del parámetro dparam	Valor del parámetro sparam
(si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_DELETE=true)				
Evento de cliquear sobre un gráfico	CHARTEVENT_CLICK	coordinada X	coordinada Y	—
Evento de cliquear sobre un objeto gráfico	CHARTEVENT_OBJECT_CLICK	coordinada X	coordinada Y	Nombre del objeto gráfico en el que ha ocurrido un evento
Evento de mover un objeto gráfico con el ratón	CHARTEVENT_OBJECT_DRAG	—	—	Nombre del objeto gráfico movido
Evento del fin de edición del texto en el campo de introducción del objeto gráfico "Campo de texto"	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nombre del objeto gráfico "Campo de texto" donde ha finalizado la introducción del texto
Evento de modificación de dimensiones del gráfico o de sus propiedades a través del diálogo de propiedades	CHARTEVENT_CHART_CHANGE	—	—	—
Identificador del evento de usuario	CHARTEVENT_CUSTOM+N	Valor determinado por la función EventChartCustom()	Valor determinado por la función EventChartCustom()	Valor determinado por la función EventChartCustom()

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#define KEY_NUMPAD_5       12
#define KEY_LEFT           37
#define KEY_UP             38
#define KEY_RIGHT          39
#define KEY_DOWN           40
#define KEY_NUMLOCK_DOWN   98
#define KEY_NUMLOCK_LEFT  100
#define KEY_NUMLOCK_5     101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP    104
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Ha sido arrancado el Asesor Experto con el nombre ",MQL5InfoString(MQL5_PROG
//--- enable object create events
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- enable object delete events
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- la actualización forzosa de las propiedades del gráfico garantiza la preparación
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identificador del evento
                  const long& lparam,   // parámetro de evento del tipo long
                  const double& dparam, // parámetro de evento del tipo double
                  const string& sparam  // parámetro de evento del tipo string
                  )
  {
//--- el botón izquierdo del ratón ha sido pulsado en el gráfico
   if(id==CHARTEVENT_CLICK)
     {
      Print("Coordinadas del clic de ratón en el gráfico: x= ",lparam,"  y= ",dparam);
     }
//--- el ratón ha hecho un clic en el objeto gráfico
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("Clic del ratón en el objeto con el nombre '"+sparam+"'");
     }
//--- la tecla del teclado ha sido apretada
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch(lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("Apretada KEY_NUMLOCK_LEFT");   break;
         case KEY_LEFT:          Print("Apretada KEY_LEFT");           break;
         case KEY_NUMLOCK_UP:    Print("Apretada KEY_NUMLOCK_UP");     break;
         case KEY_UP:            Print("Apretada KEY_UP");             break;
         case KEY_NUMLOCK_RIGHT: Print("Apretada KEY_NUMLOCK_RIGHT");  break;
         case KEY_RIGHT:         Print("Apretada KEY_RIGHT");          break;
         case KEY_NUMLOCK_DOWN:  Print("Apretada KEY_NUMLOCK_DOWN");   break;
         case KEY_DOWN:          Print("Apretada KEY_DOWN");           break;
         case KEY_NUMPAD_5:      Print("Apretada KEY_NUMPAD_5");       break;
         case KEY_NUMLOCK_5:     Print("Apretada KEY_NUMLOCK_5");      break;
         default:                Print("Apretada una de las teclas no especificadas");
        }
      ChartRedraw();
     }
//--- el objeto ha sido eliminado
   if(id==CHARTEVENT_OBJECT_DELETE)
     {
      Print("Se ha eliminado el objeto con el nombre ",sparam);
     }
//--- el objeto ha sido creado
   if(id==CHARTEVENT_OBJECT_CREATE)
     {
      Print("Se ha creado el objeto con el nombre ",sparam);
     }
//--- el objeto ha sido arrastrado o las coordinadas de puntos de anclaje han sido car
   if(id==CHARTEVENT_OBJECT_DRAG)
     {
      Print("Cambio de los puntos de enlace del objeto con el nombre ",sparam);
     }
//--- el texto en la casilla Edit del objeto gráfico ha sido cambiado
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
     {
      Print("Ha sido cambiado el texto en el objeto Edit ",sparam);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
```

For CHARTEVENT_MOUSE_MOVE event the **sparam** string parameter contains information about state of the keyboard and mouse buttons:

Bit	Description
1	State of the left mouse button
2	State of the right mouse button
3	State of the SHIFT button
4	State of the CTRL button
5	State of the middle mouse button
6	State of the first extra mouse button
7	State of the second extra mouse button

Example:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- enable CHART_EVENT_MOUSE_MOVE messages
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
// --- desactivamos el menú contextual del gráfico (a la derecha)
   ChartSetInteger(0,CHART_CONTEXT_MENU,0);
// --- desactivamos la cruz (con el botón central)
   ChartSetInteger(0,CHART_CROSSHAIR_TOOL,0);
//--- la actualización forzosa de las propiedades del gráfico garantiza la preparación
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse left
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse right
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse middle
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse first X key
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse second X key
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift key
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control key
   return(res);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

Para el evento CHARTEVENT_MOUSE_WHEEL los parámetros **lparam** y **dparam** contienen información sobre el estado de los botones **Ctrl**, **Shift**, de los botoenes del ratón, las coordenadas del cursor y la amplitud del deslizamiento de la rueda del ratón. Para que resulte más comprensible, inicie este asesor en el gráfico y deslice la rueda del ratón pulsando por turnos los diferentes botones y teclas descritas en el código.

Ejemplo de procesamiento del evento CHARTEVENT_MOUSE_WHEEL:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- activación de los mensajes sobre el deslizamiento de la rueda del ratón
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- la actualización forzosa de las propiedades del gráfico garantiza la preparación
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- vamos a analizar el estado de los botones y de la rueda del ratón para est
      int flg_keys = (int)(lparam>>32);          // bandera de estados de las teclas C
      int x_cursor = (int)(short)lparam;         // coordenada X en la que ha sucedido
      int y_cursor = (int)(short)(lparam>>16);   // coordenada Y en la que ha sucedido
      int delta    = (int)dparam;                // valor sumado del deslizamiento de
      //--- procesamos la bandera
      string str_keys="";
      if((flg_keys&0x0001)!=0) str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0) str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0) str_keys+="SHIFT ";
      if((flg_keys&0x0008)!=0) str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0) str_keys+="MMOUSE ";
```

© 2000-2025, MetaQuotes Ltd.

```
      if((flg_keys&0x0020)!=0) str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0) str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1) + "'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
     }
  }
//+------------------------------------------------------------------+ /*
   Ejemplo de muestra
   CHARTEVENT_MOUSE_WHEEL: Ctrl pressed: X=193, Y=445, delta=-120
   CHARTEVENT_MOUSE_WHEEL: Shift pressed: X=186, Y=446, delta=120
   CHARTEVENT_MOUSE_WHEEL:  X=178, Y=447, delta=-120
   CHARTEVENT_MOUSE_WHEEL:  X=231, Y=449, delta=120
   CHARTEVENT_MOUSE_WHEEL: MiddleButton pressed: X=231, Y=449, delta=120
   CHARTEVENT_MOUSE_WHEEL: LeftButton pressed: X=279, Y=320, delta=-120
   CHARTEVENT_MOUSE_WHEEL: RightButton pressed: X=253, Y=330, delta=120  */
```

Véase también

Funciones de procesamiento de eventos, Trabajo con eventos

© 2000-2025, MetaQuotes Ltd.

Periodos de gráficos

Todos los periodos de gráficos predefinidos tienen sus identificadores únicos. El identificador PERIOD_CURRENT denota el periodo corriente del gráfico en el cual se inicia el programa mql5.

ENUM_TIMEFRAMES

Identificador	Descripción
PERIOD_CURRENT	Periodo corriente
PERIOD_M1	1 minuto
PERIOD_M2	2 minutos
PERIOD_M3	3 minutos
PERIOD_M4	4 minutos
PERIOD_M5	5 minutos
PERIOD_M6	6 minutos
PERIOD_M10	10 minutos
PERIOD_M12	12 minutos
PERIOD_M15	15 minutos
PERIOD_M20	20 minutos
PERIOD_M30	30 minutos
PERIOD_H1	1 hora
PERIOD_H2	2 horas
PERIOD_H3	3 horas
PERIOD_H4	4 horas
PERIOD_H6	6 horas
PERIOD_H8	8 horas
PERIOD_H12	12 horas
PERIOD_D1	1 día
PERIOD_W1	1 semana
PERIOD_MN1	1 mes

Ejemplo:

```
   string chart_name="test_Object_Chart";
   Print("Vamos a intentar crear un objeto Chart con el nombre ",chart_name);
//--- si este objeto no existe, lo creamos
   if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- definimos el símbolo
   ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- determinamos la coordenada X para el punto de anclaje
   ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- determinamos la coordenada Y para el punto de anclaje
   ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- establecemos el ancho
   ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- establecemos el alto
   ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- definimos el periodo
   ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- determinamos la escala (de 0 a 5)
   ObjectSetDouble(0,chart_name,OBJPROP_SCALE,4);
//--- deshabilitamos la selección con el ratón
   ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

Identificadores de las series temporales

Los identificadores de las series temporales se usan en las funciones iHighest() y iLowest(). Pueden ser uno de los valores de la enumeración

ENUM_SERIESMODE

Identificador	Descripción
MODE_OPEN	Precio de apertura
MODE_LOW	Precio mínimo
MODE_HIGH	Precio máximo
MODE_CLOSE	Precio de cierre
MODE_VOLUME	Volumen de ticks
MODE_REAL_VOLUME	Volumen real
MODE_SPREAD	Spread

Véase también

PeriodSeconds, Period, Fecha y hora, Visibilidad de objetos

© 2000-2025, MetaQuotes Ltd.

Propiedades de gráficos

Los identificadores de la familia de enumeraciones ENUM_CHART_PROPERTY se usan como parámetros de las funciones para trabajar con gráfico. Las siglas r/o en la columna "Tipo de la propiedad" significa que dicha propiedad está destinada únicamente para la lectura y no puede ser cambiada. Las siglas w/o en la columna "Tipo de la propiedad" significa que dicha propiedad está destinada únicamente para la escritura y no puede ser obtenida. A la hora de acceder a algunas propiedades, es necesario indicar el parámetro-modificador adicional (modifier) que sirve para especificar el número de subventana del gráfico. 0 significa la ventana principal.

Las funciones que establecen las propiedades de los gráficos sirven principalmente para mandarle los comandos para el cambio. Cuando estas funciones se ejecuten con éxito, el comando se coloca en la cola general de los eventos del gráfico. El cambio del gráfico se realiza durante el procesamiento de la cola de eventos de este gráfico.

Por esta razón no hay que esperar la modificación visual inmediata del gráfico tras la llamada de estas funciones. En general la actualización del gráfico se realiza por el terminal de forma automática según los eventos del cambio, es decir: la llegada de una nueva cotización, cambio del tamaño de la ventana, etc.

Para la actualización forzosa de la apariencia del gráfico, se utiliza el comando de redibujo del gráfico ChartRedraw().

Para las funciones ChartSetInteger() y ChartGetInteger()

ENUM_CHART_PROPERTY_INTEGER

Identificador	Descripción	Tipo de la propiedad
CHART_SHOW	Dibujado del gráfico de precio. Si se establece false, se desactivará el dibujado de cualquier atributo del gráfico de precio y se eliminarán los bordes del gráfico: la escala de tiempo y precio, la barra de navegación rápida, los eventos del Calendario, los signos de las operaciones, las sugerencias de los indicadores y	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	barras, las subventanas de los indicadores, los histogramas de volumen, etcétera. La desactivación del dibujado es una solución ideal para crear una interfaz de programa personalizada usando los recursos gráficos Los objetos gráficos se dibujan siempre, independientemente del valor establecido en la propiedad CHART_SHOW.	
CHART_IS_OBJECT	Indicio para la identificación del objeto "Gráfico" (OBJ_CHART) - para un objeto gráfico devuelve true. Para el gráfico real devuelve false	bool r/o
CHART_BRING_TO_TOP	Visualización del gráfico por encima de los demás	bool
CHART_CONTEXT_MENU	Activar/desactivar el acceso al menú contextual pulsando el botón derecho del ratón. El valor CHART_CONTEXT_MENU=false solo desactiva el	bool (valor por defecto true)

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	menú contextual del gráfico, en este caso, además, el menú contextual para los objetos en el gráfico permanece accesible.	
CHART_CROSSHAIR_TOOL	Activar/desactivar el acceso al instrumento "Cursor en cruz" pulsando el botón medio del ratón	bool (valor por defecto true)
CHART_MOUSE_SCROLL	Desplazamiento del gráfico por la horizontal usando el botón izquierdo del ratón. Desplazamiento por la vertical también estará disponible si el valor de cualquiera de las siguientes tres propiedades está establecido en true: CHART_SCALEFIX, CHART_SCALEFIX_11 o CHART_SCALE_PT_PER_BAR Con CHART_MOUSE_SCROLL=false, el desplazamiento del gráfico con la rueda del ratón no estará disponible	bool
CHART_EVENT_MOUSE_WHEEL	Envía a todos los programas mql5	bool (valor por defecto true)

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	en el gráfico mensajes sobre los eventos de la rueda del ratón (CHARTEVENT_MOUSE_WHEEL)	
CHART_EVENT_MOUSE_MOVE	Envía una notificación sobre los eventos de desplazamiento y pulsación de los botones del ratón (CHARTEVENT_MOUSE_MOVE) a todos los programas mql5 presentes en el gráfico	bool
CHART_EVENT_OBJECT_CREATE	Envía una notificación sobre el evento de creación de nuevo objeto gráfico (CHARTEVENT_OBJECT_CREATE) a todos los programas mql5 presentes en el gráfico	bool
CHART_EVENT_OBJECT_DELETE	Envía una notificación sobre el evento de eliminación de un objeto gráfico (CHARTEVENT_OBJECT_DELETE) a todos los programas mql5 presentes en el gráfico	bool
CHART_MODE	Tipo de gráfico (velas, barras o líneas)	enum ENUM_CHART_MODE

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
CHART_FOREGROUND	Gráfico de precio en el fondo	bool
CHART_SHIFT	Modo de sangría del gráfico de precio desde el lado derecho	bool
CHART_AUTOSCROLL	Modo automático de traspaso al lado derecho del gráfico	bool
CHART_KEYBOARD_CONTROL	Autorización para controlar el gráfico usando las teclas ("Home", "End", "PageUp", "+", "-", "Flecha arriba", etcétera). La definición CHART_KEYBOARD_CONTROL=false permite desactivar el desplazamiento y el escalado del gráfico, pero manteniendo intacta la posibilidad de obtener los eventos de pulsación de estas teclas en OnChartEvent().	bool
CHART_QUICK_NAVIGATION	Permite al gráfico interceptar las teclas Space y Enter para activar la barra de navegación rápida. La barra de navegación rápida aparece de forma	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	automática debajo del gráfico al hacer doble click con el ratón o presionar las teclas Space/Enter. De este modo, es posible alternar rápidamente el símbolo, el marco temporal y la fecha de la primera barra visible.	
CHART_SCALE	Escala	int de 0 a 5
CHART_SCALEFIX	Modo de escala fija	bool
CHART_SCALEFIX_11	Modo de escala 1:1	bool
CHART_SCALE_PT_PER_BAR	Modo de especificar la escala en puntos por barra	bool
CHART_SHOW_TICKER	Representa el ticker de un símbolo en la esquina superior izquierda. Si establecemos CHART_SHOW_TICKER en el valor false, CHART_SHOW_OHLC también se establecerá temporalmente en el valor false, además de desactivarse la muestra de OHLC	bool
CHART_SHOW_OHLC	Representa los valores de OHLC en la esquina	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	superior izquierda. Si establecemos CHART_SHOW_OHLC en el valor true, CHART_SHOW_TICKER también se establecerá temporalmente en el valor true, además de activarse la muestra del ticker	
CHART_SHOW_BID_LINE	Muestra del valor Bid con una línea horizontal en el gráfico	bool
CHART_SHOW_ASK_LINE	Muestra del valor Ask con una línea horizontal en el gráfico	bool
CHART_SHOW_LAST_LINE	Muestra del valor Last con una línea horizontal en el gráfico	bool
CHART_SHOW_PERIOD_SEP	Muestra de separadores verticales entre períodos adyacentes	bool
CHART_SHOW_GRID	Muestra de rejilla en el gráfico	bool
CHART_SHOW_VOLUMES	Muestra de volúmenes en el gráfico	enum ENUM_CHART_VOLUME_MODE
CHART_SHOW_OBJECT_DESCR	Mostrar las descripciones de texto de los objetos (las descripciones no se muestran para	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	todos los objetos)	
CHART_SHOW_TRADE_HISTORY	Mostrar las transacciones de la historia comercial como flechas de entrada/salida en un gráfico. Consulte las descripciones de la opción "Mostrar historia comercial" en los ajustes de la plataforma.	
CHART_VISIBLE_BARS	Cantidad de barras en el gráfico disponibles para ser mostrados	int r/o
CHART_WINDOWS_TOTAL	Número total de las ventanas del gráfico, incluyendo las subventanas de indicadores	int r/o
CHART_WINDOW_IS_VISIBLE	Visibilidad de subventanas	bool r/o modificador - número de subventana
CHART_WINDOW_HANDLE	Manejador (handle) del gráfico (HWND)	int r/o
CHART_WINDOW_YDISTANCE	Distancia en píxeles por el eje vertical Y entre el marco superior de la subventana del indicador y el marco superior de la ventana principal del gráfico. En caso del evento del ratón, las coordinadas del	int r/o modificador - número de subventana

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	cursor se pasan en términos de las coordinadas de la ventana principal del gráfico, mientras que las coordinadas de los objetos gráficos en la subventana del indicador se establecen respecto a la esquina superior izquierda de la subventana. El valor es necesario para convertir las coordinadas absolutas del gráfico principal a las coordinadas locales de la subventana para un trabajo correcto con los objetos gráficos cuyas coordinadas se establecen respecto a la esquina superior izquierda del cuadro de la subventana.	
CHART_FIRST_VISIBLE_BAR	Número de la primera barra visible en el gráfico. Indexación de las barras corresponde a la serie temporal.	int r/o
CHART_WIDTH_IN_BARS	El ancho del gráfico en barras	int r/o

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
CHART_WIDTH_IN_PIXELS	El ancho del gráfico en píxeles	int r/o
CHART_HEIGHT_IN_PIXELS	El alto del gráfico en píxeles	int modificador - número de subventana
CHART_COLOR_BACKGROUND	Color del fondo del gráfico	color
CHART_COLOR_FOREGROUND	Color de ejes, escala y línea OHLC	color
CHART_COLOR_GRID	Color de rejilla	color
CHART_COLOR_VOLUME	Color de volúmenes y niveles de apertura de posiciones	color
CHART_COLOR_CHART_UP	Color de barra al alza, sombras y bordes del cuerpo de la vela de toro	color
CHART_COLOR_CHART_DOWN	Color de barra a la baja, sombras y bordes del cuerpo de la vela de oso	color
CHART_COLOR_CHART_LINE	Color de la línea del gráfico y de las velas japonesas "Doji"	color
CHART_COLOR_CANDLE_BULL	Color del cuerpo de la vela toro	color
CHART_COLOR_CANDLE_BEAR	Color del cuerpo de la vela oso	color
CHART_COLOR_BID	Color de la línea Bid-precio	color
CHART_COLOR_ASK	Color de la línea Ask-precio	color
CHART_COLOR_LAST	Color de la línea de precio de la última	color

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	transacción realizada (Last)	
CHART_COLOR_STOP_LEVEL	Color de los niveles de stop-órdenes (Stop Loss y Take Profit)	color
CHART_SHOW_TRADE_LEVELS	Visualiza en el gráfico los niveles comerciales (niveles de posiciones abiertas, Stop Loss, Take Profit y ordenes pendientes)	bool
CHART_DRAG_TRADE_LEVELS	Permiso para arrastrar los niveles de trading por el gráfico utilizando el ratón. Por defecto, el modo de arrastre está activado (valor true)	bool
CHART_SHOW_DATE_SCALE	Visualiza la escala de tiempo en el gráfico	bool
CHART_SHOW_PRICE_SCALE	Visualiza la escala de precios en el gráfico	bool
CHART_SHOW_ONE_CLICK	Visualización del panel del trading rápido en el gráfico (opción "Trading con un clic")	bool
CHART_IS_MAXIMIZED	La ventana del gráfico está maximizada	bool r/o
CHART_IS_MINIMIZED	La ventana del gráfico está	bool r/o

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	minimizada	
CHART_IS_DOCKED	La ventana del gráfico está fijada. Si establecemos el valor false, el gráfico se podrá arrastrar fuera de los límites del terminal	bool
CHART_FLOAT_LEFT	Coordenada izquierda del gráfico desprendido con respecto a la pantalla virtual	int
CHART_FLOAT_TOP	Coordenada superior del gráfico desprendido con respecto a la pantalla virtual	int
CHART_FLOAT_RIGHT	Coordenada derecha del gráfico desprendido con respecto a la pantalla virtual	int
CHART_FLOAT_BOTTOM	Coordenada inferior del gráfico desprendido con respecto a la pantalla virtual	int

Para las funciones ChartSetDouble() y ChartGetDouble()

ENUM_CHART_PROPERTY_DOUBLE

Identificador	Descripción	Tipo de la propiedad
CHART_SHIFT_SIZE	Tamaño de sangría de la barra cero desde el lado derecho en por cientos	double (de 10 a 50 por ciento)

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
CHART_FIXED_POSITION	Situación de la posición fija del gráfico desde el lado izquierdo en por cientos. La posición fija del gráfico está marcada con un pequeño triángulo gris sobre el eje horizontal de tiempo. Se muestra sólo si está desactivado el desplazamiento automático hacia la derecha con la llegada de un nuevo tick (ver la propiedad CHART_AUTOSCROLL). Cuando aumentamos o disminuimos el zoom, la barra que se encuentra en la posición fija se queda en el mismo sitio.	double
CHART_FIXED_MAX	Máximo fijo del gráfico	double
CHART_FIXED_MIN	Mínimo fijo del gráfico	double
CHART_POINTS_PER_BAR	Valor de la escala en puntos por barra	double
CHART_PRICE_MIN	Mínimo del gráfico	double r/o modificador - número de subventana
CHART_PRICE_MAX	Máximo del gráfico	double r/o modificador - número de subventana

Par la función ChartSetString() y ChartGetString()

ENUM_CHART_PROPERTY_STRING

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
CHART_COMMENT	Texto del comentario en el gráfico	string
CHART_EXPERT_NAME	Nombre del Expert Advisor ejecutado en el gráfico con el chart_id definido	string r/o
CHART_SCRIPT_NAME	Nombre del script ejecutado en el gráfico con el chart_id definido	string r/o

Ejemplo:

```
int chartMode=ChartGetInteger(0,CHART_MODE);
switch(chartMode)
  {
   case(CHART_BARS):    Print("CHART_BARS");   break;
   case(CHART_CANDLES): Print("CHART_CANDLES");break;
   default:Print("CHART_LINE");
  }
bool shifted=ChartGetInteger(0,CHART_SHIFT);
if(shifted) Print("CHART_SHIFT = true");
else Print("CHART_SHIFT = false");
bool autoscroll=ChartGetInteger(0,CHART_AUTOSCROLL);
if(autoscroll) Print("CHART_AUTOSCROLL = true");
else Print("CHART_AUTOSCROLL = false");
int chartHandle=ChartGetInteger(0,CHART_WINDOW_HANDLE);
Print("CHART_WINDOW_HANDLE = ",chartHandle);
int windows=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
Print("CHART_WINDOWS_TOTAL = ",windows);
if(windows>1)
  {
   for(int i=0;i<windows;i++)
     {
      int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,i);
      double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,i);
      double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,i);
      Print(i+": CHART_HEIGHT_IN_PIXELS = ",height,"pixels");
      Print(i+": CHART_PRICE_MIN = ",priceMin);
      Print(i+": CHART_PRICE_MAX = ",priceMax);
     }
  }
```

Véase también

Ejemplos de trabajo con el gráfico

© 2000-2025, MetaQuotes Ltd.

Posicionamiento de gráfico

Tres identificadores de la lista ENUM_CHART_POSITION son posibles valores del parámetro position para la función ChartNavigate().

ENUM_CHART_POSITION

Identificador	Descripción
CHART_BEGIN	Inicio del gráfico (precios más viejos)
CHART_CURRENT_POS	Posición actual
CHART_END	Fin del gráfico (precios más recientes)

Ejemplo:

```
long handle=ChartOpen("EURUSD",PERIOD_H12);
if(handle!=0)
  {
   ChartSetInteger(handle,CHART_AUTOSCROLL,false);
   ChartSetInteger(handle,CHART_SHIFT,true);
   ChartSetInteger(handle,CHART_MODE,CHART_LINE);
   ResetLastError();
   bool res=ChartNavigate(handle,CHART_END,150);
   if(!res) Print("Navigate failed. Error = ",GetLastError());
   ChartRedraw();
  }
```

© 2000-2025, MetaQuotes Ltd.

Visualización de gráficos

Los gráficos de precios se puede representar de tres maneras:

- como barras;
- como velas;
- como línea quebrada.

El tipo concreto de mostrar el gráfico de precio se establece por la función ChartSetInteger(handle_de_gráfico,CHART_MODE, tipo_de_gráfico), donde el tipo_de_gráfico es uno de los valores de enumeración ENUM_CHART_MODE.

ENUM_CHART_MODE

Identificador	Descripción
CHART_BARS	Representación en forma de barras
CHART_CANDLES	Representación en forma de velas japonesas
CHART_LINE	Representación en forma de una línea trazada por los precios Close

Para especificar el régimen de mostrar volúmenes en el gráfico de precio utilizan la función ChartSetInteger(handle_de_gráfico, CHART_SHOW_VOLUMES, tipo_de_mostrar), donde tipo_de_mostrar es uno de los valores de enumeración ENUM_CHART_VOLUME_MODE.

ENUM_CHART_VOLUME_MODE

Identificador	Descripción
CHART_VOLUME_HIDE	Volúmenes no se muestran
CHART_VOLUME_TICK	Volúmenes de tick
CHART_VOLUME_REAL	Volúmenes comerciales

Ejemplo:

```
//--- obtenemos manejador (handle) del gráfico corriente
   long handle=ChartID();
   if(handle>0) // en caso de éxito, personalizamos más
     {
      //--- deshabilitamos desplazamiento automático
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- fijamos la sangría del lado derecho del gráfico
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- representamos en forma de velas
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- desplazamos a 100 barras desde el inicio de historial
      ChartNavigate(handle,CHART_CURRENT_POS,100);
      //--- establecemos el modo de mostrar volúmenes de tick
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
     }
```

© 2000-2025, MetaQuotes Ltd.

Véase también

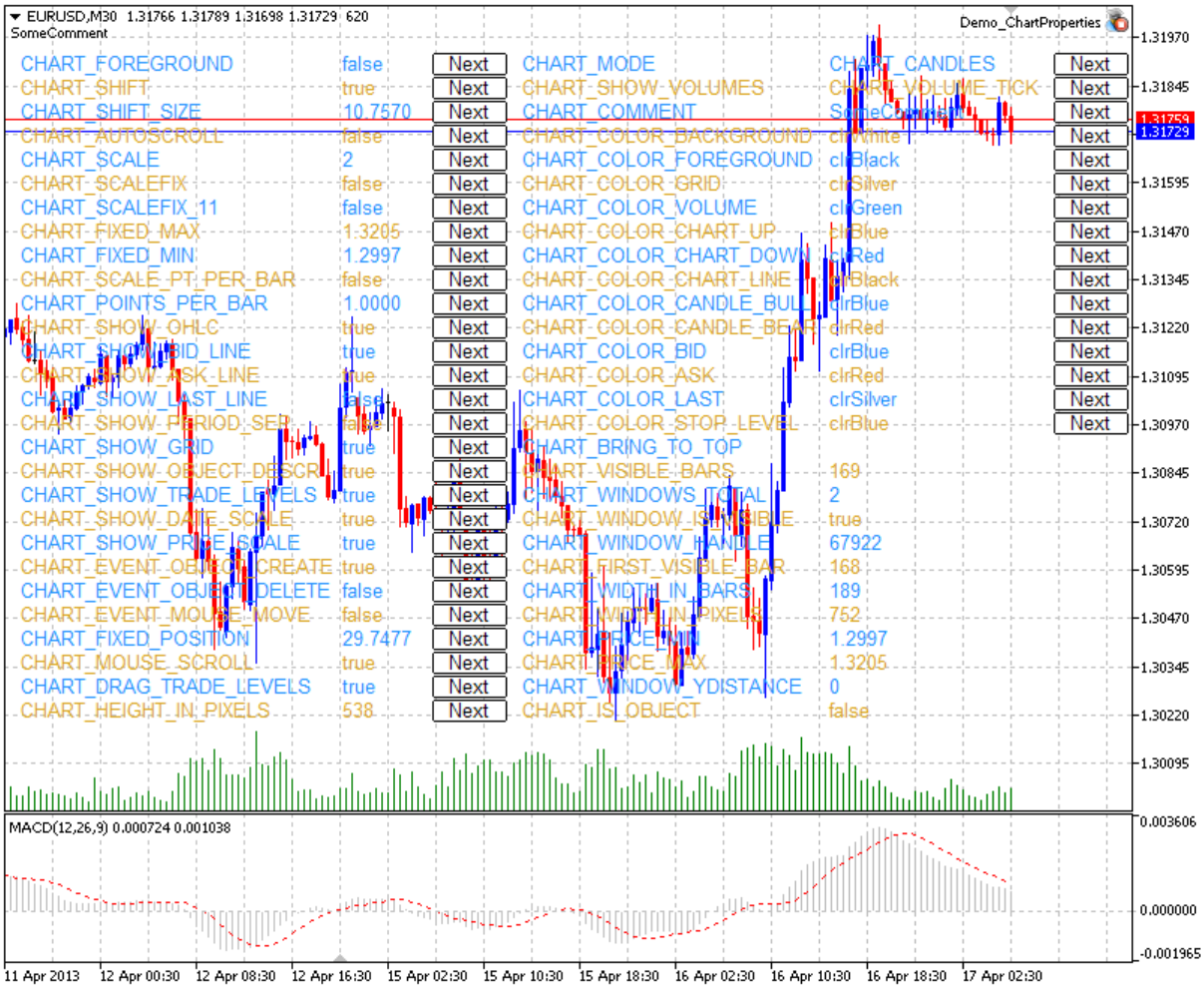
ChartOpen, ChartID

© 2000-2025, MetaQuotes Ltd.

Ejemplos de trabajo con el gráfico

Este apartado contiene varios ejemplos para el trabajo con las propiedades del gráfico. Para cada propiedad se exponen una o dos funciones terminadas que permiten establecer / obtener el valor de esta propiedad. Puede utilizar estas funciones en sus programas MQL5 tal como están.

En la imagen de abajo se ve un panel gráfico que demuestra de forma muy clara cómo el cambio de una propiedad del gráfico cambia su apariencia. El hacer click en el botón "Next" permite establecer un valor nuevo para la propiedad correspondiente y ver los cambios en la ventana del gráfico que eso provoca.



El código fuente de este panel se encuentra más abajo.

Propiedades del gráfico y ejemplos de las funciones para trabajar con ellas

- **CHART_IS_OBJECT** - determina si el objeto es un gráfico real o un objeto gráfico.

```
//+------------------------------------------------------------------+
//| Determinar que si el objeto es un gráfico. Si es                 |
//| un objeto gráfico, el resultado es true. Si es un                |
//| gráfico real, entonces la variable result obtiene el valor false.|
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ChartIsObject(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos la propiedad del gráfico
   if(!ChartGetInteger(chart_ID,CHART_IS_OBJECT,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      //--- devolvemos false
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_BRING_TO_TOP** - muestra el gráfico por encima de los demás.

```
//+------------------------------------------------------------------+
//| Enviar al terminal los comandos para mostrar el gráfico          |
//| por encima de los demás.                                         |
//+------------------------------------------------------------------+
bool ChartBringToTop(const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- mostramos el gráfico por encima de los demás
   if(!ChartSetInteger(chart_ID,CHART_BRING_TO_TOP,0,true))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_MOUSE_SCROLL** - la propiedad de desplazamiento del gráfico con el botón izquierdo del ratón.

```
//+------------------------------------------------------------------+
//| La función determina si se puede desplazar el gráfico utilizando |
```

© 2000-2025, MetaQuotes Ltd.

```
//| el botón izquierdo del ratón.                                    |
//+------------------------------------------------------------------+
bool ChartMouseScrollGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de desplazamiento          |
//| del gráfico con el botón izquierdo del ratón.                    |
//+------------------------------------------------------------------+
bool ChartMouseScrollSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_EVENT_MOUSE_MOVE** - la propiedad del envío de los mensajes sobre los eventos de desplazamiento y pulsación de los botones del ratón a los programas mql5 (CHARTEVENT_MOUSE_MOVE).

```
//+------------------------------------------------------------------+
//| Comprobar si se envían los mensajes sobre los eventos            |
//| de desplazamiento y pulsación de los botones del ratón           |
//| en este gráfico a los programas mql5.                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ChartEventMouseMoveGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo del envío de los mensajes  |
//| sobre los eventos de desplazamiento y pulsación de los botones   |
//| del ratón a todos los programas mql5 en este gráfico.            |
//+------------------------------------------------------------------+
bool ChartEventMouseMoveSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_EVENT_OBJECT_CREATE** - la propiedad del envío de los mensajes sobre el evento de creación de un objeto gráfico a los programas mql5 (CHARTEVENT_OBJECT_CREATE).

```
//+------------------------------------------------------------------+
//| Comprobar si se envían los mensajes sobre el evento de creación  |
//| del objeto gráfico a todos los programas mql5 en este gráfico.   |
//+------------------------------------------------------------------+
bool ChartEventObjectCreateGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
```

© 2000-2025, MetaQuotes Ltd.

```
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo del envío de los mensajes  |
//| sobre el evento de creación de un objeto gráfico a todos         |
//| los programas mql5 en este gráfico.                              |
//+------------------------------------------------------------------+
bool ChartEventObjectCreateSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_EVENT_OBJECT_DELETE** - la propiedad del envío de los mensajes sobre el evento de eliminación de un objeto gráfico a los programas mql5 (CHARTEVENT_OBJECT_DELETE).

```
//+------------------------------------------------------------------+
//| Comprobar si se envían los mensajes sobre el evento              |
//| de eliminación del objeto gráfico a todos los programas          |
//| mql5 en este gráfico.                                            |
//+------------------------------------------------------------------+
bool ChartEventObjectDeleteGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo del envío de los mensajes  |
//| sobre el evento de eliminación de un objeto gráfico a todos      |
//| los programas mql5 en este gráfico.                              |
//+------------------------------------------------------------------+
bool ChartEventObjectDeleteSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_MODE** - tipo del gráfico (velas, barras u línea).

```
//+------------------------------------------------------------------+
//| Obtener el tipo de visualización del gráfico (en forma de velas, |
//| barras o línea).                                                 |
//+------------------------------------------------------------------+
ENUM_CHART_MODE ChartModeGet(const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=WRONG_VALUE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_MODE,0,result))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((ENUM_CHART_MODE)result);
  }
//+------------------------------------------------------------------+
//| Establecer el tipo de visualización del gráfico (en forma de     |
//| velas, barras o línea).                                          |
//+------------------------------------------------------------------+
bool ChartModeSet(const long value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_MODE,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_FOREGROUND** - la propiedad de visualización del gráfico de precios en el primer plano.

```
//+------------------------------------------------------------------+
//| La función determina si se visualiza el gráfico de precios en    |
//| el primer plano.                                                 |
//+------------------------------------------------------------------+
bool ChartForegroundGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización del       |
//| gráfico de precios en el primer plano.                           |
//+------------------------------------------------------------------+
bool ChartForegroundSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHIFT** - el modo de desplazamiento del gráfico de precios desde el borde derecho.

```
//+------------------------------------------------------------------+
//| La función determina si está activado el modo de visualización   |
//| del gráfico de precios con el desplazamiento desde               |
//|  el borde derecho.                                               |
//+------------------------------------------------------------------+
bool ChartShiftGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización del       |
//| gráfico de precios con el desplazamiento desde el borde derecho. |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ChartShiftSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_AUTOSCROLL** - el modo de desplazamiento automático hacia el borde derecho del gráfico.

```
//+------------------------------------------------------------------+
//| La función determina si está activado el modo de desplazamiento  |
//| automático del gráfico hacia la derecha cuando se reciben        |
//| nuevos ticks.                                                    |
//+------------------------------------------------------------------+
bool ChartAutoscrollGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de desplazamiento          |
//| automático del gráfico hacia la derecha cuando se reciben        |
//| nuevos ticks.                                                    |
//+------------------------------------------------------------------+
bool ChartAutoscrollSet(const bool value,const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SCALE** - la propiedad de la escala del gráfico.

```
//+------------------------------------------------------------------+
//| Obtener la escala del gráfico (de 0 a 5).                        |
//+------------------------------------------------------------------+
int ChartScaleGet(const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=-1;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SCALE,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((int)result);
  }
//+------------------------------------------------------------------+
//| Establecer la escala del gráfico (de 0 a 5).                     |
//+------------------------------------------------------------------+
bool ChartScaleSet(const long value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SCALE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
```

- **CHART_SCALEFIX** - el modo de la escala fija del gráfico.

```
//+------------------------------------------------------------------+
//| La función determina si está activado el modo de la escala       |
//| fija del gráfico.                                                |
//+------------------------------------------------------------------+
bool ChartScaleFixGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de la escala fija.         |
//+------------------------------------------------------------------+
bool ChartScaleFixSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SCALEFIX_11** - el modo de la escala del gráfico 1:1.

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| La función determina si está activado el modo de la escala "1:1".|
//+------------------------------------------------------------------+
bool ChartScaleFix11Get(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de la escala "1:1"         |
//+------------------------------------------------------------------+
bool ChartScaleFix11Set(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SCALE_PT_PER_BAR** - el modo de especificación de la escala del gráfico en puntos por barra.

```
//+------------------------------------------------------------------+
//| La función determina si está activado el modo de especificación  |
//| de la escala en puntos por barra.                                |
//+------------------------------------------------------------------+
bool ChartScalePerBarGet(bool &result,const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de especificación de la    |
//| escala en puntos por barra.                                      |
//+------------------------------------------------------------------+
bool ChartScalePerBarSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHOW_OHLC** - la propiedad de visualización de los valores OHLC en la esquina superior izquierda.

```
//+------------------------------------------------------------------+
//| La función determina si está activado el modo de visualización de|
//| los valores OHLC en la esquina superior izquierda.               |
//+------------------------------------------------------------------+
bool ChartShowOHLCGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización de los    |
//| valores OHLC en la esquina superior izquierda del gráfico.       |
//+------------------------------------------------------------------+
bool ChartShowOHLCSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHOW_BID_LINE** - la propiedad de visualización del valor Bid como una línea horizontal en el gráfico.

```
//+------------------------------------------------------------------+
//| La función determina si está activado el modo de visualización de|
//| la línea Bid en el gráfico.                                      |
//+------------------------------------------------------------------+
bool ChartShowBidLineGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización de la     |
//| línea Bid en el gráfico.                                         |
//+------------------------------------------------------------------+
bool ChartShowBidLineSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHOW_ASK_LINE** - la propiedad de visualización del valor Ask como una línea horizontal en el gráfico.

```
//+------------------------------------------------------------------+
//| La función determina si está activado el modo de visualización   |
//| de la línea Ask en el gráfico.                                   |
//+------------------------------------------------------------------+
bool ChartShowAskLineGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización de la     |
//| línea Ask en el gráfico.                                         |
//+------------------------------------------------------------------+
bool ChartShowAskLineSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHOW_LAST_LINE** - la propiedad de visualización del valor Last como una línea horizontal en el gráfico.

```
//+------------------------------------------------------------------+
//| La función determina si está activado el modo de visualización   |
//| de la línea para el precio de la última transacción realizada.   |
//+------------------------------------------------------------------+
bool ChartShowLastLineGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización de la     |
//| línea del precio de la última transacción realizada.             |
//+------------------------------------------------------------------+
bool ChartShowLastLineSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHOW_PERIOD_SEP** - la propiedad de visualización de los separadores verticales entre los períodos adyacentes.

```
//+------------------------------------------------------------------+
//| La función determina si está activado el modo de visualización   |
//| de los separadores verticales entre los períodos adyacentes.     |
//+------------------------------------------------------------------+
bool ChartShowPeriodSeparatorGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización de        |
//| los separadores verticales entre los períodos adyacentes.        |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ChartShowPeriodSepapatorSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHOW_GRID** - la propiedad de visualización de cuadrícula en el gráfico.

```
//+------------------------------------------------------------------+
//| La función determina si la cuadrícula se visualiza en el gráfico.|
//+------------------------------------------------------------------+
bool ChartShowGridGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva la visualización de la cuadrícula  |
//| en el gráfico.                                                   |
//+------------------------------------------------------------------+
bool ChartShowGridSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_GRID,0,value))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- mostramos el mensaje del error en el diario "Asesores Expertos"
       Print(__FUNCTION__+", Error Code = ",GetLastError());
       return(false);
      }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHOW_VOLUMES** - la propiedad de visualización de volúmenes en el gráfico.

```
//+------------------------------------------------------------------+
//| La función determina si se visualizan los volúmenes en el gráfico|
//| (no se muestran, se muestran los de ticks, se muestran           |
//| los reales).                                                     |
//+------------------------------------------------------------------+
ENUM_CHART_VOLUME_MODE ChartShowVolumesGet(const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=WRONG_VALUE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_VOLUMES,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((ENUM_CHART_VOLUME_MODE)result);
  }
//+------------------------------------------------------------------+
//| La función establece el modo de visualización de los volúmenes   |
//| en el gráfico.                                                   |
//+------------------------------------------------------------------+
bool ChartShowVolumesSet(const long value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_VOLUMES,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
```

- **CHART_SHOW_OBJECT_DESCR** - la propiedad de las descripciones emergentes de los objetos gráficos.

```
//+------------------------------------------------------------------+
//| La función determina si se visualizan las descripciones          |
//| emergentes de los objetos gráficos al situar el cursor           |
//| sobre ellos.                                                     |
//+------------------------------------------------------------------+
bool ChartShowObjectDescriptionGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización de las    |
//| descripciones emergentes de los objetos gráficos al situar       |
//|  el cursor sobre ellos.                                          |
//+------------------------------------------------------------------+
bool ChartShowObjectDescriptionSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_VISIBLE_BARS** - determina el número de barras en el gráfico que están disponibles para la visualización.

```
//+------------------------------------------------------------------+
//| La función obtiene el número de barras que se muestran           |
//| (están visibles) en la ventana del gráfico.                      |
//+------------------------------------------------------------------+
int ChartVisibleBars(const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=-1;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_VISIBLE_BARS,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((int)result);
  }
```

- **CHART_WINDOWS_TOTAL** - determina el número total de las ventanas del gráfico, incluyendo las subventanas de los indicadores.

```
//+------------------------------------------------------------------+
//| La función obtiene el número total de las ventanas del gráfico,  |
//| incluyendo las subventanas de los indicadores.                   |
//+------------------------------------------------------------------+
int ChartWindowsTotal(const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=-1;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_WINDOWS_TOTAL,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((int)result);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_WINDOW_IS_VISIBLE** - determina la visibilidad de la subventana.

```
//+------------------------------------------------------------------+
//| La función determina si esta ventana o subventana del gráfico es |
//| visible.                                                         |
//+------------------------------------------------------------------+
bool ChartWindowsIsVisible(bool &result,const long chart_ID=0,const int sub_window=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_IS_VISIBLE,sub_window,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_WINDOW_HANDLE** - devuelve el manejador del gráfico.

```
//+------------------------------------------------------------------+
//| La función obtiene el manejador del gráfico                      |
//+------------------------------------------------------------------+
int ChartWindowsHandle(const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=-1;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_HANDLE,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((int)result);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_WINDOW_YDISTANCE** - determina la distancia en píxeles entre el marco superior de la subventana del indicador y el marco superior de la ventana principal del gráfico.

```
//+------------------------------------------------------------------+
//| La función obtiene la distancia en píxeles entre el marco        |
//| superior de la subventana y el marco superior de la ventana      |
//| principal del gráfico.                                           |
//+------------------------------------------------------------------+
int ChartWindowsYDistance(const long chart_ID=0,const int sub_window=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=-1;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_YDISTANCE,sub_window,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((int)result);
  }
```

- **CHART_FIRST_VISIBLE_BAR** - devuelve el número de la primera barra visible en el gráfico (la indexación de las barras corresponde a la serie temporal).

```
//+------------------------------------------------------------------+
//| La función obtiene el número de la primera barra visible en      |
//| el gráfico. La indexación se realiza como en una serie temporal, |
//| las últimas barras tienen los índices menores.                   |
//+------------------------------------------------------------------+
int ChartFirstVisibleBar(const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=-1;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_FIRST_VISIBLE_BAR,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((int)result);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
```

- **CHART_WIDTH_IN_BARS** - devuelve el ancho del gráfico en barras.

```
//+------------------------------------------------------------------+
//| La función obtiene el valor del ancho del gráfico en barras.     |
//+------------------------------------------------------------------+
int ChartWidthInBars(const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=-1;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_BARS,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((int)result);
  }
```

- **CHART_WIDTH_IN_PIXELS** - devuelve el ancho del gráfico en píxeles.

```
//+------------------------------------------------------------------+
//| La función obtiene el valor del ancho del gráfico en píxeles.    |
//+------------------------------------------------------------------+
int ChartWidthInPixels(const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=-1;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_PIXELS,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((int)result);
  }
```

- **CHART_HEIGHT_IN_PIXELS** - la propiedad del alto del gráfico en píxeles.

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| La función obtiene el valor del alto del gráfico en píxeles.     |
//+------------------------------------------------------------------+
int ChartHeightInPixelsGet(const long chart_ID=0,const int sub_window=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long result=-1;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((int)result);
  }
//+------------------------------------------------------------------+
//| La función establece el valor del alto del gráfico en píxeles.   |
//+------------------------------------------------------------------+
bool ChartHeightInPixelsSet(const int value,const long chart_ID=0,const int sub_windo
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_BACKGROUND** - color del fondo del gráfico.

```
//+------------------------------------------------------------------+
//| La función obtiene el color del fondo del gráfico.               |
//+------------------------------------------------------------------+
color ChartBackColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color del fondo del gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BACKGROUND,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La función establece el color del fondo del gráfico.             |
//+------------------------------------------------------------------+
bool ChartBackColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color del fondo del gráfico
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BACKGROUND,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_FOREGROUND** - el color de los ejes, escala y la línea OHLC.

```
//+------------------------------------------------------------------+
//| La función obtiene el color de los ejes, escala y la línea       |
//| OHLC del gráfico.                                                |
//+------------------------------------------------------------------+
color ChartForeColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color de los ejes, escala y la línea OHLC del gráfico
   if(!ChartGetInteger(chart_ID,CHART_COLOR_FOREGROUND,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| La función establece el color de los ejes, escala y la línea     |
//| OHLC del gráfico.                                                 |
//+------------------------------------------------------------------+
bool ChartForeColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color de los ejes, escala y la línea OHLC del gráfico
   if(!ChartSetInteger(chart_ID,CHART_COLOR_FOREGROUND,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_GRID** - el color de la cuadrícula del gráfico.

```
//+------------------------------------------------------------------+
//| La función obtiene el color de la cuadrícula del gráfico.        |
//+------------------------------------------------------------------+
color ChartGridColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color de la cuadrícula del gráfico
   if(!ChartGetInteger(chart_ID,CHART_COLOR_GRID,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La función establece el color de la cuadrícula del gráfico.      |
//+------------------------------------------------------------------+
bool ChartGridColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color de la cuadrícula del gráfico
   if(!ChartSetInteger(chart_ID,CHART_COLOR_GRID,clr))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_VOLUME** - el color de los volúmenes y niveles de la apertura de posiciones.

```
//+------------------------------------------------------------------+
//| La función obtiene el color de visualización de los volúmenes    |
//| y niveles de la apertura de posiciones.                          |
//+------------------------------------------------------------------+
color ChartVolumeColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color de volúmenes y niveles de la apertura de posiciones
   if(!ChartGetInteger(chart_ID,CHART_COLOR_VOLUME,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La función establece el color de visualización de los volúmenes  |
//| y niveles de la apertura de posiciones.                          |
//+------------------------------------------------------------------+
bool ChartVolumeColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color de volúmenes y niveles de la apertura de posiciones
   if(!ChartSetInteger(chart_ID,CHART_COLOR_VOLUME,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_COLOR_CHART_UP** - el color de la barra arriba, sombra y borde del cuerpo de vela alcista.

```
//+------------------------------------------------------------------+
//| La función obtiene el color de la barra que va arriba, el color  |
//| de la sombra y del borde del cuerpo de vela alcista.             |
//+------------------------------------------------------------------+
color ChartUpColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color de la barra arriba, sombra y borde del cuerpo de la vela alci
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_UP,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La función establece el color de la barra que va arriba, el color|
//| de la sombra y del borde del cuerpo de vela alcista.             |
//+------------------------------------------------------------------+
bool ChartUpColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color de la barra arriba, sombra y borde del cuerpo de la vela a
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_UP,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_CHART_DOWN** - el color de la barra abajo, sombra y borde del cuerpo de vela bajista.

```
//+------------------------------------------------------------------+
//| La función obtiene el color de la barra que va abajo, el color de|
//| la sombra y del borde del cuerpo de vela bajista.                |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
color ChartDownColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color de la barra abajo, sombra y borde del cuerpo de la vela bajis
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_DOWN,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La función establece el color de la barra que va abajo, el color |
//| de la sombra y del borde del cuerpo de vela bajista.             |
//+------------------------------------------------------------------+
bool ChartDownColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color de la barra abajo, sombra y borde del cuerpo de la vela ba
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_DOWN,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_CHART_LINE** - el color de la línea del gráfico y de las velas japonesas "Doji".

```
//+------------------------------------------------------------------+
//| La función obtiene el color de la línea del gráfico y de las     |
//| velas japonesas "Doji".                                          |
//+------------------------------------------------------------------+
color ChartLineColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color de la línea del gráfico y de las velas japonesas "Doji"
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_LINE,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La función establece el color de la línea del gráfico y de las   |
//| velas japonesas "Doji".                                          |
//+------------------------------------------------------------------+
bool ChartLineColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color de la línea del gráfico y de las velas japonesas "Doji"
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_LINE,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BULL** - el color del cuerpo de la vela alcista.

```
//+------------------------------------------------------------------+
//| La función obtiene el color del cuerpo de la vela alcista.       |
//+------------------------------------------------------------------+
color ChartBullColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color del cuerpo de la vela alcista
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| La función establece el color del cuerpo de la vela alcista.     |
//+------------------------------------------------------------------+
bool ChartBullColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color del cuerpo de la vela alcista
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BEAR** - el color del cuerpo de la vela bajista.

```
//+------------------------------------------------------------------+
//| La función obtiene el color del cuerpo de la vela bajista.       |
//+------------------------------------------------------------------+
color ChartBearColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color del cuerpo de la vela bajista
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La función establece el color del cuerpo de la vela bajista.     |
//+------------------------------------------------------------------+
bool ChartBearColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color del cuerpo de la vela bajista
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_BID** - el color de la línea del precio Bid.

```
//+------------------------------------------------------------------+
//| La función obtiene el color de la línea del precio Bid.          |
//+------------------------------------------------------------------+
color ChartBidColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color de la línea del precio Bid
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BID,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La función establece el color de la línea del precio Bid.        |
//+------------------------------------------------------------------+
bool ChartBidColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color de la línea del precio Bid
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BID,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_ASK** - el color de la línea del precio Ask.

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| La función obtiene el color de la línea del precio Ask.          |
//+------------------------------------------------------------------+
color ChartAskColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color de la línea del precio Ask
   if(!ChartGetInteger(chart_ID,CHART_COLOR_ASK,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La función establece el color de la línea del precio Ask.        |
//+------------------------------------------------------------------+
bool ChartAskColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color de la línea del precio Ask
   if(!ChartSetInteger(chart_ID,CHART_COLOR_ASK,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_LAST** - el color de la línea del precio de la última transacción realizada (Last).

```
//+------------------------------------------------------------------+
//| La función obtiene el color de la línea del precio de la última  |
//| transacción realizada.                                           |
//+------------------------------------------------------------------+
color ChartLastColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- obtenemos el color de la línea del precio de la última transacción realizada (La
   if(!ChartGetInteger(chart_ID,CHART_COLOR_LAST,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La función establece el color de la línea del precio de la última|
//| transacción realizada.                                           |
//+------------------------------------------------------------------+
bool ChartLastColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color de la línea del precio de la última transacción realizada
   if(!ChartSetInteger(chart_ID,CHART_COLOR_LAST,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_COLOR_STOP_LEVEL** - el color de los niveles de las órdenes Stop (Stop Loss y Take Profit).

```
//+------------------------------------------------------------------+
//| La función obtiene el color de los niveles Stop Loss             |
//| y Take Profit.                                                   |
//+------------------------------------------------------------------+
color ChartStopLevelColorGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el color
   long result=clrNONE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el color de los niveles de las órdenes Stop (Stop Loss y Take Profit)
   if(!ChartGetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return((color)result);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| La función establece el color de los niveles Stop Loss y         |
//| Take Profit.                                                     |
//+------------------------------------------------------------------+
bool ChartStopLevelColorSet(const color clr,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el color de los niveles de las órdenes Stop (Stop Loss y Take Profi
   if(!ChartSetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,clr))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHOW_TRADE_LEVELS** - la propiedad de visualización de los niveles de trad¡ng en el gráfico (niveles de posiciones abiertas, Stop Loss, Take Profit y órdenes pendientes).

```
//+------------------------------------------------------------------+
//| La función determina si los niveles de trading se visualizan     |
//| en el gráfico.                                                   |
//+------------------------------------------------------------------+
bool ChartShowTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización de los    |
//| niveles de trading.                                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ChartShowTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_DRAG_TRADE_LEVELS** - la propiedad del permiso para arrastrar los niveles de trading por el gráfico utilizando el ratón.

```
//+------------------------------------------------------------------+
//| La función determina si se puede arrastrar los niveles de trading|
//| en el gráfico utilizando el ratón.                               |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de arrastre de los niveles |
//| de trading en el gráfico utilizando el ratón.                    |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHOW_DATE_SCALE** - la propiedad de visualización de la escala de tiempo en el gráfico.

```
//+------------------------------------------------------------------+
//| La función determina si la escala de tiempo se visualiza         |
//| en el gráfico.                                                   |
//+------------------------------------------------------------------+
bool ChartShowDateScaleGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización de la     |
//| escala de tiempo en el gráfico.                                  |
//+------------------------------------------------------------------+
bool ChartShowDateScaleSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_SHOW_PRICE_SCALE** - la propiedad de visualización de la escala de precios en el gráfico.

```
//+------------------------------------------------------------------+
//| La función determina si la escala de precios se visualiza        |
//| en el gráfico.                                                   |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa / desactiva el modo de visualización de la     |
//| escala de precios en el gráfico.                                 |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleSet(const bool value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
```

- **CHART_SHOW_ONE_CLICK** - visualización del panel del trading rápido en el gráfico (opción "Trading con un clic").

```
//+------------------------------------------------------------------+
//| La función determina si se visualiza el panel del                |
//| trading rápido en el gráfico (opción "Trading con un clic")      |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelGet(bool &result,const long chart_ID=0)
  {
//--- preparamos la variable para obtener el valor de la propiedad
   long value;
//--- anulamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función activa/desactiva el modo de visualización del         |
//| panel de trading rápido en el gráfico                            |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelSet(const bool value,const long chart_ID=0)
  {
//--- anulamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_SHIFT_SIZE** - el tamaño de desplazamiento de la barra cero desde el borde derecho en por cientos.

```
//+------------------------------------------------------------------+
//| La función obtiene el tamaño de desplazamiento de la barra cero  |
//| desde el borde derecho del gráfico en por cientos (de 10% a 50%).|
//+------------------------------------------------------------------+
double ChartShiftSizeGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el resultado
   double result=EMPTY_VALUE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetDouble(chart_ID,CHART_SHIFT_SIZE,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return(result);
  }
//+------------------------------------------------------------------+
//| La función establece el tamaño de desplazamiento de la barra cero|
//| desde el borde derecho del gráfico en por cientos (de 10% a 50%).|
//| Para activar el modo de desplazamiento, hay que establecer el    |
//| valor de la propiedad CHART_SHIFT igual atrue.                   |
//+------------------------------------------------------------------+
bool ChartShiftSizeSet(const double value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetDouble(chart_ID,CHART_SHIFT_SIZE,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_FIXED_POSITION** - la posición fija del gráfico desde el borde izquierdo en por cientos.

```
//+------------------------------------------------------------------+
//| La función obtiene la posición fija del gráfico desde            |
//| el borde izquierdo en por cientos.                               |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
double ChartFixedPositionGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el resultado
   double result=EMPTY_VALUE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetDouble(chart_ID,CHART_FIXED_POSITION,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return(result);
  }
//+------------------------------------------------------------------+
//| La función establece la posición fija del gráfico desde          |
//| el borde izquierdo en por cientos. Para ver la                   |
//| posición fija en el gráfico, antes hay que establecer            |
//| el valor de la propiedad CHART_AUTOSCROLL igual a false.         |
//+------------------------------------------------------------------+
bool ChartFixedPositionSet(const double value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetDouble(chart_ID,CHART_FIXED_POSITION,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_FIXED_MAX** - la propiedad del máximo fijo del gráfico.

```
//+------------------------------------------------------------------+
//| La función obtiene el valor del máximo fijo del gráfico.         |
//+------------------------------------------------------------------+
double ChartFixedMaxGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el resultado
   double result=EMPTY_VALUE;
//--- reseteamos el valor del error
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- obtenemos el valor de la propiedad
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MAX,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return(result);
  }
//+------------------------------------------------------------------+
//| La función establece el valor del máximo fijo del gráfico.       |
//| Para poder modificar el valor de esta propiedad, hace falta      |
//| establecer antes el valor de la propiedad CHART_SCALEFIX igual a |
//| true.                                                            |
//+------------------------------------------------------------------+
bool ChartFixedMaxSet(const double value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MAX,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_FIXED_MIN** - la propiedad del mínimo fijo del gráfico.

```
//+------------------------------------------------------------------+
//| La función obtiene el valor del mínimo fijo del gráfico.         |
//+------------------------------------------------------------------+
double ChartFixedMinGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el resultado
   double result=EMPTY_VALUE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MIN,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
   return(result);
  }
//+------------------------------------------------------------------+
//| La función establece el valor del mínimo fijo del gráfico.       |
//| Para poder modificar el valor de esta propiedad, hace falta      |
//| establecer antes el valor de la propiedad CHART_SCALEFIX igual a |
//| true.                                                            |
//+------------------------------------------------------------------+
bool ChartFixedMinSet(const double value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MIN,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_POINTS_PER_BAR** - el valor de la escala en puntos por barra.

```
//+------------------------------------------------------------------+
//| La función obtiene el valor de la escala del gráfico             |
//| en puntos por barra.                                             |
//+------------------------------------------------------------------+
double ChartPointsPerBarGet(const long chart_ID=0)
  {
//--- preparamos la variable para recibir el resultado
   double result=EMPTY_VALUE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetDouble(chart_ID,CHART_POINTS_PER_BAR,0,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return(result);
  }
//+------------------------------------------------------------------+
//| La función establece el valor de la escala del gráfico en puntos |
//| por barra. Para ver el resultado del cambio del valor de esta    |
//| propiedad, antes hay que establecer el valor de la propiedad     |
```

© 2000-2025, MetaQuotes Ltd.

```
//| CHART_SCALE_PT_PER_BAR igual a true.                             |
//+------------------------------------------------------------------+
bool ChartPointsPerBarSet(const double value,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetDouble(chart_ID,CHART_POINTS_PER_BAR,value))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_PRICE_MIN** - devuelve el valor del mínimo del gráfico.

```
//+------------------------------------------------------------------+
//| La función obtiene el valor del mínimo del gráfico en la ventana |
//| principal o subventana.                                          |
//+------------------------------------------------------------------+
double ChartPriceMin(const long chart_ID=0,const int sub_window=0)
  {
//--- preparamos la variable para recibir el resultado
   double result=EMPTY_VALUE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MIN,sub_window,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return(result);
  }
```

- **CHART_PRICE_MAX** - devuelve el valor del máximo del gráfico.

```
//+------------------------------------------------------------------+
//| La función obtiene el valor del máximo del gráfico en la ventana |
//| principal o subventana.                                          |
//+------------------------------------------------------------------+
double ChartPriceMax(const long chart_ID=0,const int sub_window=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- preparamos la variable para recibir el resultado
   double result=EMPTY_VALUE;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MAX,sub_window,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- devolvemos el valor de la propiedad del gráfico
   return(result);
  }
```

- **CHART_COMMENT** - texto del comentario en el gráfico.

```
//+------------------------------------------------------------------+
//| La función obtiene el texto del comentario en la esquina superior|
//| izquierda del gráfico.                                           |
//+------------------------------------------------------------------+
bool ChartCommentGet(string &result,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetString(chart_ID,CHART_COMMENT,result))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función establece el texto del comentario en la esquina       |
//| superior izquierda del gráfico.                                  |
//+------------------------------------------------------------------+
bool ChartCommentSet(const string str,const long chart_ID=0)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- establecemos el valor de la propiedad
   if(!ChartSetString(chart_ID,CHART_COMMENT,str))
     {
      //--- mostramos el mensaje del error en el diario "Asesores Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- ejecución con éxito
   return(true);
  }
```

- **CHART_IS_MAXIMIZED** - окно графика развернуто

```
//+------------------------------------------------------------------+
//| La función determina si la ventana actual del gráfico            |
//| está maximizada                                                  |
//+------------------------------------------------------------------+
bool ChartWindowsIsMaximized(bool &result,const long chart_ID=0)
  {
//--- preparando la variable para obtener el valor de las propiedades
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_IS_MAXIMIZED))
     {
      //--- mostramos un mensaje sobre el error en el diario de registro "Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecutado con éxito
   return(true);
  }
```

- **CHART_IS_MINIMIZED** - la ventana del gráfico está minimizada

```
//+------------------------------------------------------------------+
//| La función determina si la ventana actual del gráfico            |
```

© 2000-2025, MetaQuotes Ltd.

```
//| está minimizada                                                  |
//+------------------------------------------------------------------+
bool ChartWindowsIsMinimized(bool &result,const long chart_ID=0)
  {
//--- preparando la variable para obtener el valor de las propiedades
   long value;
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el valor de la propiedad
   if(!ChartGetInteger(chart_ID,CHART_IS_MINIMIZED))
     {
      //--- mostramos un mensaje sobre el error en el diario de registro "Expertos"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- guardamos en la variable el valor de la propiedad del gráfico
   result=value;
//--- ejecutado con éxito
   return(true);
  }
```

Panel para las propiedades del gráfico

```
//--- conectamos la biblioteca de los elementos de control
#include <ChartObjects\ChartObjectsTxtControls.mqh>
//--- constantes predefinidas
#define X_PROPERTY_NAME_1    10  // coordenada X del nombre de la propiedad en la prir
#define X_PROPERTY_VALUE_1   225 // x-coordenada X del valor de la propiedad en la pri
#define X_PROPERTY_NAME_2    345 // coordenada X del nombre de la propiedad en la segu
#define X_PROPERTY_VALUE_2   550 // coordenada X del valor de la propiedad en la segur
#define X_BUTTON_1           285 // coordenada X del botón en la primera columna
#define X_BUTTON_2           700 // coordenada X del botón en la segunda columna
#define Y_PROPERTY_1         30  // coordenada Y del inicio de la primera y la segunda
#define Y_PROPERTY_2         286 // coordenada Y del inicio de la tercera columna
#define Y_DISTANCE           16  // distancia por el eje Y entre las filas
#define LAST_PROPERTY_NUMBER 111 // el número de la última propiedad gráfica
//--- parámetros de entrada
input color InpFirstColor=clrDodgerBlue; // Color de las filas impares
input color InpSecondColor=clrGoldenrod; // Color de las filas pares
//--- variables y arrays
CChartObjectLabel  ExtLabelsName[];  // etiquetas para visualizar los nombres de propi
CChartObjectLabel  ExtLabelsValue[]; // etiquetas para visualizar los valores de propi
CChartObjectButton ExtButtons[];     // botones
int                ExtNumbers[];     // índices de propiedades
string             ExtNames[];       // nombres de propiedades
uchar              ExtDataTypes[];   // tipos de datos de propiedades (integer, double
uint               ExtGroupTypes[];  // array que almacena los datos sobre la pertener
uchar              ExtDrawTypes[];   // array que almacena los datos sobre el modo de
double             ExtMaxValue[];    // valores máximos permitidos para las propiedade
double             ExtMinValue[];    // valores mínimos permitidos para las propiedade
double             ExtStep[];        // pasos para cambiar las propiedades
```

© 2000-2025, MetaQuotes Ltd.

```
int                ExtCount;         // número total de todas las propiedades
color              ExtColors[2];     // array de colores para visualizar las filas
string             ExtComments[2];   // array de comentarios (para la propiedad CHART_
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mostramos el comentario sobre el gráfico
   Comment("SomeComment");
//--- guardamos los colores en el array para conmutarse luego entre ellos
   ExtColors[0]=InpFirstColor;
   ExtColors[1]=InpSecondColor;
//--- guardamos los comentarios en el array para conmutarse luego entre ellos
   ExtComments[0]="FirstComment";
   ExtComments[1]="SecondComment";
//--- preparamos y visualizamos el panel de control de las propiedades del gráfico
   if(!PrepareControls())
      return(INIT_FAILED);
//--- ejecución con éxito
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Deinitialization function of the expert                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- quitamos el texto del comentario en el gráfico
   Comment("");
  }
//+------------------------------------------------------------------+
//| Manejador de eventos del gráfico                                 |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- comprobación del evento de pinchar sobre un objeto del gráfico
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      //--- dividimos el nombre del objeto por el separador
      string obj_name[];
      StringSplit(sparam,'_',obj_name);
      //--- prueba de que si el objeto es un botón
      if(obj_name[0]=="Button")
        {
         //--- obtenemos el índice del botón
         int index=(int)StringToInteger(obj_name[1]);
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- ponemos el botón en el estado no presionado
         ExtButtons[index].State(false);
         //--- establecemos nuevo valor de la propiedad en función de su tipo
         if(ExtDataTypes[index]=='I')
            ChangeIntegerProperty(index);
         if(ExtDataTypes[index]=='D')
            ChangeDoubleProperty(index);
         if(ExtDataTypes[index]=='S')
            ChangeStringProperty(index);
        }
     }
//--- redibujo de valores de propiedad
   RedrawProperties();
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Cambio de la propiedad integer del gráfico                       |
//+------------------------------------------------------------------+
void ChangeIntegerProperty(const int index)
  {
//--- obtenemos el valor actual de la propiedad
   long value=ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index]);
//--- determinamos el siguiente valor de la propiedad
   switch(ExtDrawTypes[index])
     {
      case 'C':
         value=GetNextColor((color)value);
         break;
      default:
         value=(long)GetNextValue((double)value,index);
         break;
     }
//--- establecemos nuevo valor de la propiedad
   ChartSetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index],0,value);
  }
//+------------------------------------------------------------------+
//| Cambio de la propiedad double del gráfico                        |
//+------------------------------------------------------------------+
void ChangeDoubleProperty(const int index)
  {
//--- obtenemos el valor actual de la propiedad
   double value=ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index]);
//--- determinamos el siguiente valor de la propiedad
   value=GetNextValue(value,index);
//--- establecemos nuevo valor de la propiedad
   ChartSetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index],value);
  }
//+------------------------------------------------------------------+
//| Cambio de la propiedad string del gráfico                        |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeStringProperty(const int index)
  {
//--- variable estática para conmutar dentro del array de comentarios ExtComments
   static uint comment_index=1;
//--- cambiamos el índice para obtener otro comentario
   comment_index=1-comment_index;
//--- establecemos nuevo valor de la propiedad
   ChartSetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[index],ExtComments[comment_
  }
//+------------------------------------------------------------------+
//| Determinación del siguiente valor de la propiedad                |
//+------------------------------------------------------------------+
double GetNextValue(const double value,const int index)
  {
   if(value+ExtStep[index]<=ExtMaxValue[index])
      return(value+ExtStep[index]);
   else
      return(ExtMinValue[index]);
  }
//+------------------------------------------------------------------+
//| Obtención del siguiente color para la propiedad del tipo color   |
//+------------------------------------------------------------------+
color GetNextColor(const color clr)
  {
//--- devolvemos el siguiente valor del color
   switch(clr)
     {
      case clrWhite: return(clrRed);
      case clrRed:   return(clrGreen);
      case clrGreen: return(clrBlue);
      case clrBlue:  return(clrBlack);
      default:       return(clrWhite);
     }
  }
//+------------------------------------------------------------------+
//| Redibujo de valores de las propiedades                           |
//+------------------------------------------------------------------+
void RedrawProperties(void)
  {
//--- texto del valor de la propiedad
   string text;
   long   value;
//--- ciclo según el número de propiedades
   for(int i=0;i<ExtCount;i++)
     {
      text="";
      switch(ExtDataTypes[i])
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         case 'I':
            //--- obtenemos el valor actual de la propiedad
            if(!ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[i],0,value))
            break;
            //--- texto de la propiedad integer
            switch(ExtDrawTypes[i])
              {
               //--- propiedad del color
               case 'C':
                  text=(string)((color)value);
                  break;
                  //--- propiedad booleana
               case 'B':
                  text=(string)((bool)value);
                  break;
                  //--- propiedad de la enumeración ENUM_CHART_MODE
               case 'M':
                  text=EnumToString((ENUM_CHART_MODE)value);
                  break;
                  //--- propiedad de la enumeración ENUM_CHART_VOLUME_MODE
               case 'V':
                  text=EnumToString((ENUM_CHART_VOLUME_MODE)value);
                  break;
                  //--- número del tipo int
               default:
                  text=IntegerToString(value);
                  break;
              }
            break;
         case 'D':
            //--- texto de la propiedad double
            text=DoubleToString(ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumber
            break;
         case 'S':
            //--- texto de la propiedad string
            text=ChartGetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[i]);
            break;
        }
      //--- visualizamos el valor de la propiedad
      ExtLabelsValue[i].Description(text);
     }
  }
//+------------------------------------------------------------------+
//| Crear el panel para gestionar las propiedades del gráfico        |
//+------------------------------------------------------------------+
bool PrepareControls(void)
  {
//--- adjudicamos la memoria para los arrays con reserva
   MemoryAllocation(LAST_PROPERTY_NUMBER+1);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- variables
   int i=0;      // variable del ciclo
   int col_1=0; // número de propiedades en la primera columna
   int col_2=0; // número de propiedades en la segunda columna
   int col_3=0; // número de propiedades en la tercera columna
//--- número de propiedades actuales - 0
   ExtCount=0;
//--- buscamos las propiedades en el ciclo
   while(i<=LAST_PROPERTY_NUMBER)
     {
      //--- guardamos el número actual de la propiedad
      ExtNumbers[ExtCount]=i;
      //--- aumentamos el valor de la variable del ciclo
      i++;
      //--- comprobamos si hay propiedad con este número
      if(CheckNumber(ExtNumbers[ExtCount],ExtNames[ExtCount],ExtDataTypes[ExtCount],Ex
        {
         //--- creamos los elementos de control para la propiedad
         switch(ExtGroupTypes[ExtCount])
           {
            case 1:
               //--- creamos las etiquetas y el botón para la propiedad
               if(!ShowProperty(ExtCount,0,X_PROPERTY_NAME_1,X_PROPERTY_VALUE_1,X_BUTT
               return(false);
               //--- el número de elementos en la primera columna se ha aumentado
               col_1++;
               break;
            case 2:
               //--- creamos las etiquetas y el botón para la propiedad
               if(!ShowProperty(ExtCount,1,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,X_BUTT
               return(false);
               //--- el número de elementos en la segunda columna se ha aumentado
               col_2++;
               break;
            case 3:
               //--- creamos sólo las etiquetas para la propiedad
               if(!ShowProperty(ExtCount,2,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,0,Y_PR
               return(false);
               //--- el número de elementos en la tercera columna se ha aumentado
               col_3++;
               break;
           }
         //--- determinamos el valor máximo, mínimo y el paso
         GetMaxMinStep(ExtNumbers[ExtCount],ExtMaxValue[ExtCount],ExtMinValue[ExtCount
         //--- aumentamos el número de propiedades
         ExtCount++;
        }
     }
//--- liberamos la memoria que no se utiliza por los arrays
```

© 2000-2025, MetaQuotes Ltd.

```
   MemoryAllocation(ExtCount);
//--- redibujamos los valores de las propiedades
   RedrawProperties();
   ChartRedraw();
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Adjudicación de memoria para los arrays                          |
//+------------------------------------------------------------------+
void MemoryAllocation(const int size)
  {
   ArrayResize(ExtLabelsName,size);
   ArrayResize(ExtLabelsValue,size);
   ArrayResize(ExtButtons,size);
   ArrayResize(ExtNumbers,size);
   ArrayResize(ExtNames,size);
   ArrayResize(ExtDataTypes,size);
   ArrayResize(ExtGroupTypes,size);
   ArrayResize(ExtDrawTypes,size);
   ArrayResize(ExtMaxValue,size);
   ArrayResize(ExtMinValue,size);
   ArrayResize(ExtStep,size);
  }
//+------------------------------------------------------------------+
//| Comprobamos si el índice de la propiedad pertenece a una de      |
//| las enumeraciones ENUM_CHART_PROPERTIES                          |
//+------------------------------------------------------------------+
bool CheckNumber(const int ind,string &name,uchar &data_type,uint &group_type,uchar &d
  {
//--- comprobamos si la propiedad es del tipo entero (integer)
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_INTEGER)ind);
   if(_LastError==0)
     {
      data_type='I';                       // propiedad de la enumeración ENUM_CHART_PR
      GetTypes(ind,group_type,draw_type); // definimos los parámetros de visualizació
      return(true);
     }
//--- comprobamos si la propiedad es del tipo real (double)
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_DOUBLE)ind);
   if(_LastError==0)
     {
      data_type='D';                       // propiedad de la enumeración ENUM_CHART_PR
      GetTypes(ind,group_type,draw_type); // definimos los parámetros de visualizació
      return(true);
     }
//--- comprobamos si la propiedad es del tipo literal (string)
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_STRING)ind);
   if(_LastError==0)
     {
      data_type='S';                        // propiedad de la enumeración ENUM_CHART_PR
      GetTypes(ind,group_type,draw_type); // definimos los parámetros de visualizació
      return(true);
     }
//--- la propiedad no pertenece a ninguna de las enumeraciones
   return(false);
  }
//+------------------------------------------------------------------+
//| Determinación del grupo en el que debe almacenarse la propiedad,     |
//| así como su tipo de visualización                                    |
//+------------------------------------------------------------------+
void GetTypes(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- comprobamos si la propiedad pertenece al tercer grupo
//--- las propiedades del tercer grupo se muestran en la segunda columna empezando de
   if(CheckThirdGroup(property_number,group_type,draw_type))
      return;
//--- comprobamos si la propiedad pertenece al segundo grupo
//--- las propiedades del segundo grupo se muestran en la segunda columna desde su ini
   if(CheckSecondGroup(property_number,group_type,draw_type))
      return;
//--- si nos hemos encontrado aquí, entonces la propiedad pertenece al primer grupo (p
   CheckFirstGroup(property_number,group_type,draw_type);
  }
//+------------------------------------------------------------------+
//| La función comprueba si la propiedad pertenece al tercer grupo, y     |
//| si es así, determina su tipo de visualización                        |
//+------------------------------------------------------------------+
bool CheckThirdGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- comprobamos si la propiedad pertenece al tercer grupo
   switch(property_number)
     {
      //--- propiedades booleanas
      case CHART_IS_OBJECT:
      case CHART_WINDOW_IS_VISIBLE:
         draw_type='B';
         break;
         //--- propiedades enteras (integer)
      case CHART_VISIBLE_BARS:
      case CHART_WINDOWS_TOTAL:
      case CHART_WINDOW_HANDLE:
      case CHART_WINDOW_YDISTANCE:
      case CHART_FIRST_VISIBLE_BAR:
      case CHART_WIDTH_IN_BARS:
```

© 2000-2025, MetaQuotes Ltd.

```
      case CHART_WIDTH_IN_PIXELS:
         draw_type='I';
         break;
         //--- propiedades reales (double)
      case CHART_PRICE_MIN:
      case CHART_PRICE_MAX:
         draw_type='D';
         break;
         //--- de hecho, esta propiedad es un comando de visualización del gráfico por
         //--- no hay necesidad de aplicarla para este panel, porque la ventana siempr
         //--- va a colocarse por encima de todas las demás antes de que la utilicemos
      case CHART_BRING_TO_TOP:
         draw_type=' ';
         break;
         //--- la propiedad no pertenece al tercer grupo
      default:
         return(false);
     }
//--- la propiedad pertenece al tercer grupo
   group_type=3;
   return(true);
  }
//+------------------------------------------------------------------+
//| La función comprueba si la propiedad pertenece al segundo grupo, y si |
//| es así, determina su tipo de visualización                       |
//+------------------------------------------------------------------+
bool CheckSecondGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- comprobamos si la propiedad pertenece al segundo grupo
   switch(property_number)
     {
      //--- la propiedad del tipo ENUM_CHART_MODE
      case CHART_MODE:
         draw_type='M';
         break;
         //--- la propiedad del tipo ENUM_CHART_VOLUME_MODE
      case CHART_SHOW_VOLUMES:
         draw_type='V';
         break;
         //--- propiedad literal (string)
      case CHART_COMMENT:
         draw_type='S';
         break;
         //--- propiedad del color
      case CHART_COLOR_BACKGROUND:
      case CHART_COLOR_FOREGROUND:
      case CHART_COLOR_GRID:
      case CHART_COLOR_VOLUME:
      case CHART_COLOR_CHART_UP:
```

© 2000-2025, MetaQuotes Ltd.

```
      case CHART_COLOR_CHART_DOWN:
      case CHART_COLOR_CHART_LINE:
      case CHART_COLOR_CANDLE_BULL:
      case CHART_COLOR_CANDLE_BEAR:
      case CHART_COLOR_BID:
      case CHART_COLOR_ASK:
      case CHART_COLOR_LAST:
      case CHART_COLOR_STOP_LEVEL:
         draw_type='C';
         break;
         //--- la propiedad no pertenece al segundo grupo
      default:
         return(false);
     }
//--- la propiedad pertenece al segundo grupo
   group_type=2;
   return(true);
  }
//+------------------------------------------------------------------+
//| Esta función se invoca sólo si ya se sabe   |
//| que la propiedad no pertenece ni al segundo, ni al tercer grupo de propiedades
//+------------------------------------------------------------------+
void CheckFirstGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- la propiedad pertenece al primer grupo
   group_type=1;
//--- determinamos el tipo de visualización de la propiedad
   switch(property_number)
     {
      //--- propiedades enteras (integer)
      case CHART_SCALE:
      case CHART_HEIGHT_IN_PIXELS:
         draw_type='I';
         return;
         //--- propiedades reales (double)
      case CHART_SHIFT_SIZE:
      case CHART_FIXED_POSITION:
      case CHART_FIXED_MAX:
      case CHART_FIXED_MIN:
      case CHART_POINTS_PER_BAR:
         draw_type='D';
         return;
         //--- sólo quedan las propiedades booleanas
      default:
         draw_type='B';
         return;
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Crear las etiquetas y el botón para la propiedad                         |
//+------------------------------------------------------------------+
bool ShowProperty(const int ind,const int type,const int x1,const int x2,
                  const int xb,const int y,const bool btn)
  {
//--- array estático para conmutar dentro del array de color ExtColors
   static uint color_index[3]={1,1,1};
//--- cambiamos el índice para obtener otro color
   color_index[type]=1-color_index[type];
//--- mostramos etiquetas y el botón (si btn=true) para la propiedad
   if(!LabelCreate(ExtLabelsName[ind],"name_"+(string)ind,ExtNames[ind],ExtColors[colo
      return(false);
   if(!LabelCreate(ExtLabelsValue[ind],"value_"+(string)ind,"",ExtColors[color_index[t
      return(false);
   if(btn && !ButtonCreate(ExtButtons[ind],(string)ind,xb,y+1))
      return(false);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Crear etiqueta                                                   |
//+------------------------------------------------------------------+
bool LabelCreate(CChartObjectLabel &lbl,const string name,const string text,
                 const color clr,const int x,const int y)
  {
   if(!lbl.Create(0,"Label_"+name,0,x,y)) return(false);
   if(!lbl.Description(text))             return(false);
   if(!lbl.FontSize(10))                  return(false);
   if(!lbl.Color(clr))                    return(false);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Crear el botón                                                   |
//+------------------------------------------------------------------+
bool ButtonCreate(CChartObjectButton &btn,const string name,
                  const int x,const int y)
  {
   if(!btn.Create(0,"Button_"+name,0,x,y,50,15)) return(false);
   if(!btn.Description("Next"))                  return(false);
   if(!btn.FontSize(10))                         return(false);
   if(!btn.Color(clrBlack))                      return(false);
   if(!btn.BackColor(clrWhite))                  return(false);
   if(!btn.BorderColor(clrBlack))                return(false);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Establecemos el valor máximo, mínimo y el paso  |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void GetMaxMinStep(const int property_number,double &max,double &min,double &step)
  {
   double value;
//--- establecemos valores en función del tipo de la propiedad
   switch(property_number)
     {
      case CHART_SCALE:
         max=5;
         min=0;
         step=1;
         break;
      case CHART_MODE:
      case CHART_SHOW_VOLUMES:
         max=2;
         min=0;
         step=1;
         break;
      case CHART_SHIFT_SIZE:
         max=50;
         min=10;
         step=2.5;
         break;
      case CHART_FIXED_POSITION:
         max=90;
         min=0;
         step=15;
         break;
      case CHART_POINTS_PER_BAR:
         max=19;
         min=1;
         step=3;
         break;
      case CHART_FIXED_MAX:
         value=ChartGetDouble(0,CHART_FIXED_MAX);
         max=value*1.25;
         min=value;
         step=value/32;
         break;
      case CHART_FIXED_MIN:
         value=ChartGetDouble(0,CHART_FIXED_MIN);
         max=value;
         min=value*0.75;
         step=value/32;
         break;
      case CHART_HEIGHT_IN_PIXELS:
         max=700;
         min=520;
         step=30;
```

© 2000-2025, MetaQuotes Ltd.

```
         break;
         //--- valores por defecto
      default:
         max=1;
         min=0;
         step=1;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Constantes de objetos

Hay 44 objetos gráficos que se puede crear y mostrar en el gráfico de precios. Todas las constantes destinadas para trabajar con objetos están divididas en 9 grupos:

- Tipos de objetos – identificadores de objetos gráficos;
- Propiedades de objetos – trabajo con las propiedades de objetos gráficos;
- Modos de enlace de objetos – constantes de posicionamiento de objetos en el gráfico;
- Esquina de enlace – permite fijar la esquina del gráfico respecto a la cual se realiza el posicionamiento del objeto en píxeles;
- Visibilidad de objetos – establecimiento de períodos en los que un objeto está visible;
- Niveles de las ondas de Elliot – gradación de la marcación ondulada;
- Objetos de Gann – constantes de tendencia para el abanico de Gann y la retícula de Gann;
- Colores Web – constantes de los colores Web predefinidos;
- Wingdings – códigos de caracteres de la fuente Wingdings.

© 2000-2025, MetaQuotes Ltd.

Tipos de objetos

Cuando la función ObjectCreate() crea un objeto gráfico, es necesario especificar el tipo del objeto que se va a crear, el cual podrá adquirir uno de los valores de la enumeración ENUM_OBJECT. Las siguientes especificaciones de las propiedades del objeto creado son posibles mediante las funciones para el trabajo con objetos gráficos.

ENUM_OBJECT

Identificador		Descripción
OBJ_VLINE		Línea vertical
OBJ_HLINE		Línea horizontal
OBJ_TREND		Línea de tendencia
OBJ_TRENDBYANGLE		Línea de tendencia por ángulo
OBJ_CYCLES		Líneas cíclicas
OBJ_ARROWED_LINE		Objeto "Línea de separación con una flecha"
OBJ_CHANNEL		Canal equidistante
OBJ_STDDEVCHANNEL		Canal de desviación estándar
OBJ_REGRESSION		Canal en regresión lineal
OBJ_PITCHFORK		Tridente de Andrews
OBJ_GANNLINE		Líneas de Gann
OBJ_GANNFAN		Abanico de Gann
OBJ_GANNGRID		Cuadrícula de Gann
OBJ_FIBO		Retrocesos de Fibonacci
OBJ_FIBOTIMES		Zonas temporales de Fibonacci
OBJ_FIBOFAN		Abanico de Fibonacci
OBJ_FIBOARC		Arcos de Fibonacci
OBJ_FIBOCHANNEL		Canal de Fibonacci
OBJ_EXPANSION		Expansión de Fibonacci
OBJ_ELLIOTWAVE5		Ondas de Elliott de fase impulsiva (5)
OBJ_ELLIOTWAVE3		Ondas de Elliott de fase correctiva (3)
OBJ_RECTANGLE		Rectángulo
OBJ_TRIANGLE		Triángulo
OBJ_ELLIPSE		Elipse
OBJ_ARROW_THUMB_UP		Signo "Bien" (dedo pulgar arriba)

© 2000-2025, MetaQuotes Ltd.

Identificador		Descripción
OBJ_ARROW_THUMB_DOWN		Signo "Mal" (dedo pulgar abajo)
OBJ_ARROW_UP		Signo "Flecha arriba"
OBJ_ARROW_DOWN		Signo "Flecha abajo"
OBJ_ARROW_STOP		Signo "Stop"
OBJ_ARROW_CHECK		Signo "Marca" (tic)
OBJ_ARROW_LEFT_PRICE		Etiqueta izquierda de precio
OBJ_ARROW_RIGHT_PRICE		Etiqueta derecha de precio
OBJ_ARROW_BUY		Signo "Buy"
OBJ_ARROW_SELL		Signo "Sell"
OBJ_ARROW		Objeto "Flecha"
OBJ_TEXT		Objeto "Texto"
OBJ_LABEL		Objeto "Etiqueta de texto"
OBJ_BUTTON		Objeto "Botón"
OBJ_CHART		Objeto "Gráfico"
OBJ_BITMAP		Objeto "Dibujo"
OBJ_BITMAP_LABEL		Objeto "Etiqueta gráfica"
OBJ_EDIT		Objeto "Campo de edición"
OBJ_EVENT		Objeto "Evento" corresponde a un evento en el calendario económico
OBJ_RECTANGLE_LABEL		Objeto "Etiqueta rectangular" sirve para crear y formatear la interfaz gráfica personalizada.

© 2000-2025, MetaQuotes Ltd.

OBJ_VLINE

Linea vertical.



Nota

Cuando se crea una línea vertical, se puede indicar el modo de visualización de la línea en todas las ventanas del gráfico (propiedad OBJPROP_RAY).

Ejemplo

El siguiente script crea y desplaza una línea vertical en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Línea vertical\"."
#property description "La fecha del punto de anclaje se establece en por cientos del a
#property description "de la ventana del gráfico en barras."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string           InpName="VLine";      // Nombre de la línea
input int              InpDate=25;           // Fecha de la línea en %
input color            InpColor=clrRed;      // Color de la línea
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH;  // Estilo de la línea
input int              InpWidth=3;           // Grosor de la línea
input bool             InpBack=false;        // Línea al fondo
input bool             InpSelection=true;    // Seleccionar para mover
input bool             InpRay=true;          // Continuación de la línea abajo
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpHidden=true;      // Ocultar en la lista de objetos
input long            InpZOrder=0;         // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crear línea vertical                                             |
//+------------------------------------------------------------------+
bool VLineCreate(const long            chart_ID=0,        // ID del gráfico
                 const string          name="VLine",      // nombre de la línea
                 const int             sub_window=0,      // índice de subventana
                 datetime              time=0,            // hora de la línea
                 const color           clr=clrRed,        // color de la línea
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de la línea
                 const int             width=1,           // grosor de la línea
                 const bool            back=false,        // al fondo
                 const bool            selection=true,    // seleccionar para mover
                 const bool            ray=true,          // continuación de la línea
                 const bool            hidden=true,       // ocultar en la lista de ob
                 const long            z_order=0)         // prioridad para el clic de
  {
//--- si la hora de la línea no está definida, la trazamos en la última barra
   if(!time)
      time=TimeCurrent();
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la línea vertical
   if(!ObjectCreate(chart_ID,name,OBJ_VLINE,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la línea vertical! Código del error = ",GetLastError());
      return(false);
     }
//--- establecemos el color de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de visualización de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la línea con ra
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activar (true) o desactivar (false) el modo de visualización de la línea en las
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY,ray);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover línea vertical                                             |
//+------------------------------------------------------------------+
bool VLineMove(const long   chart_ID=0,   // ID del gráfico
               const string name="VLine", // nombre de la línea
               datetime     time=0)       // hora de la línea
  {
//--- si la hora de la línea no está definida, la movemos a la última barra
   if(!time)
      time=TimeCurrent();
//--- anulamos el valor del error
   ResetLastError();
//--- movemos la línea vertical
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la línea vertical! Código del error = ",GetLastError())
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Eliminar línea vertical                                          |
//+------------------------------------------------------------------+
bool VLineDelete(const long   chart_ID=0,   // ID del gráfico
                 const string name="VLine") // nombre de la línea
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la línea vertical
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la línea vertical! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- array para guardar los valores de las fechas que van a utilizarse
//--- para establecer y cambiar las coordenadas del punto de anclaje de la línea
   datetime date[];
//--- asignación de la memoria
   ArrayResize(date,bars);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- definimos puntos para trazar la línea
   int d=InpDate*(bars-1)/100;
//--- creamos la línea vertical
   if(!VLineCreate(0,InpName,0,date[d],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpRay,InpHidden,InpZOrder))
      return;
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover la línea
//--- contador del ciclo
   int h_steps=bars/2;
//--- movemos la línea
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d<bars-1)
         d+=1;
      //--- movemos el punto
      if(!VLineMove(0,InpName,date[d]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,03 segundo
      Sleep(30);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el canal desde el gráfico
   VLineDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_HLINE

Línea horizontal.



Ejemplo

El siguiente script crea y desplaza una línea horizontal en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Línea horizontal\"."
#property description "El precio del punto de anclaje se establece en por cientos del
#property description "de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="HLine";      // Nombre de la línea
input int             InpPrice=25;          // Precio de la línea en %
input color           InpColor=clrRed;      // Color de la línea
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Estilo de la línea
input int             InpWidth=3;           // Grosor de la línea
input bool            InpBack=false;        // Línea al fondo
input bool            InpSelection=true;    // Seleccionar para mover
input bool            InpHidden=true;       // Ocultar en la lista de objetos
input long            InpZOrder=0;          // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crear línea horizontal                                           |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool HLineCreate(const long            chart_ID=0,        // ID del gráfico
                 const string          name="HLine",      // nombre de la línea
                 const int             sub_window=0,      // índice de subventana
                 double                price=0,           // precio de la línea
                 const color           clr=clrRed,        // color de la línea
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de la línea
                 const int             width=1,           // grosor de la línea
                 const bool            back=false,        // al fondo
                 const bool            selection=true,    // seleccionar para mover
                 const bool            hidden=true,       // ocultar en la lista de ob
                 const long            z_order=0)         // prioridad para el clic de
  {
//--- si el precio no ha sido establecido, la ponemos en el nivel del precio Bid actua
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la línea horizontal
   if(!ObjectCreate(chart_ID,name,OBJ_HLINE,sub_window,0,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la línea horizontal! Código del error = ",GetLastError
      return(false);
     }
//--- establecemos el color de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de visualización de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la línea con ra
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Desplazamiento de la línea horizontal                            |
//+------------------------------------------------------------------+
bool HLineMove(const long   chart_ID=0,   // ID del gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
                const string name="HLine", // nombre de la línea
                double       price=0)      // precio de la línea
  {
//--- si el precio de la línea no ha sido establecido, la movemos en el nivel del prec
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos la línea horizontal
   if(!ObjectMove(chart_ID,name,0,0,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la línea horizontal! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Eliminar la línea horizontal                                     |
//+------------------------------------------------------------------+
bool HLineDelete(const long   chart_ID=0,   // ID del gráfico
                 const string name="HLine") // nombre de la línea
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la línea horizontal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la línea horizontal! Código del error = ",GetLastErr
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpPrice<0 || InpPrice>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- tamaño del array price
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- array para guardar los valores de los precios que van a utilizarse
//--- para establecer y cambiar las coordenadas del punto de anclaje de la línea
   double price[];
//--- asignación de la memoria
   ArrayResize(price,accuracy);
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos puntos para trazar la línea
   int p=InpPrice*(accuracy-1)/100;
//--- creamos la línea horizontal
   if(!HLineCreate(0,InpName,0,price[p],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover la línea
//--- contador del ciclo
   int v_steps=accuracy/2;
//--- movemos la línea
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p<accuracy-1)
         p+=1;
      //--- movemos el punto
      if(!HLineMove(0,InpName,price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos del gráfico
   HLineDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
```

© 2000-2025, MetaQuotes Ltd.

```
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TREND

Línea de tendencia.



Nota

Para la línea de tendencia se puede indicar el modo de su continuación a la derecha y/o a la izquierda (propiedades OBJPROP_RAY_RIGHT y OBJPROP_RAY_LEFT respectivamente).

Ejemplo

El siguiente script crea y desplaza la línea de tendencia en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Línea de tendencia\"."
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="Trend";     // Nombre de la línea
input int             InpDate1=35;         // Fecha del 1-er punto en %
input int             InpPrice1=60;        // Precio del 1-er punto en %
input int             InpDate2=65;         // Fecha del 2-do punto en %
input int             InpPrice2=40;        // Precio del 2-do punto en %
input color           InpColor=clrRed;     // Color de la línea
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Estilo de la línea
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpWidth=2;          // Grosor de la línea
input bool             InpBack=false;       // Línea al fondo
input bool             InpSelection=true;   // Seleccionar para mover
input bool             InpRayLeft=false;    // Continuación de la línea a la izquierda
input bool             InpRayRight=false;   // Continuación de la línea a la derecha
input bool             InpHidden=true;      // Ocultar en la lista de objetos
input long             InpZOrder=0;         // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crear la línea de tendencia según las coordenadas establecidas
//+------------------------------------------------------------------+
bool TrendCreate(const long            chart_ID=0,        // ID del gráfico
                 const string          name="TrendLine",  // Nombre de la línea
                 const int             sub_window=0,      // índice de subventana
                 datetime              time1=0,           // hora del primer punto
                 double                price1=0,          // precio del primer punto
                 datetime              time2=0,           // hora del segundo punto
                 double                price2=0,          // precio del segundo punto
                 const color           clr=clrRed,        // color de la línea
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de la línea
                 const int             width=1,           // grosor de la línea
                 const bool            back=false,        // al fondo
                 const bool            selection=true,    // seleccionar para mover
                 const bool            ray_left=false,    // continuación de la línea
                 const bool            ray_right=false,   // continuación de la línea
                 const bool            hidden=true,       // ocultar en la lista de ob
                 const long            z_order=0)         // prioridad para el clic de
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeTrendEmptyPoints(time1,price1,time2,price2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la línea de tendencia según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_TREND,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la línea de tendencia! Código del error = ",GetLastErro
      return(false);
     }
//--- establecemos el color de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de visualización de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la línea con ra
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
```

© 2000-2025, MetaQuotes Ltd.

```
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activamos (true) o desactivamos (false) el modo de continuación de la línea a la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activamos (true) o desactivamos (false) el modo de continuación de la línea a la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover el punto de anclaje de la línea de tendencia               |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // ID del gráfico
                      const string name="TrendLine", // nombre de la línea
                      const int    point_index=0,    // número del punto de anclaje
                      datetime     time=0,           // coordenada del tiempo del punt
                      double       price=0)          // coordenada del precio del punt
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje de la línea de tendencia
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función elimina la línea de tendencia desde el gráfico.       |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // ID del gráfico
                 const string name="TrendLine") // nombre de la línea
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la línea de tendencia
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la línea de tendencia! Código del error = ",GetLastE
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje de la línea y para      |
//| los valores vacíos establece los valores por defecto                     |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- si la hora del primer punto no ha sido establecida, se colocará en la barra actu
   if(!time1)
      time1=TimeCurrent();
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del segundo punto no ha sido establecida, se colocará a 9 barras a la
   if(!time2)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- colocamos el segundo punto a 9 barras a la izquierda del primero
      time2=temp[0];
     }
//--- si el precio del segundo punto no ha sido establecido, va a coincidir con el pre
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje de la línea
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos puntos para trazar la línea
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- creamos la línea de tendencia
   if(!TrendCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,InpStyle
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje de la línea
//--- contador del ciclo
   int v_steps=accuracy/5;
//--- movemos el primer punto de anclaje verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1>1)
         p1-=1;
      //--- movemos el punto
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
       //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
       if(IsStopped())
         return;
       //--- redibujamos el gráfico
       ChartRedraw();
      }
//--- movemos el segundo punto de anclaje verticalmente
   for(int i=0;i<v_steps;i++)
     {
       //--- cogemos el siguiente valor
       if(p2<accuracy-1)
         p2+=1;
       //--- movemos el punto
       if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
       //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
       if(IsStopped())
         return;
       //--- redibujamos el gráfico
       ChartRedraw();
      }
//--- retardo de medio segundo
   Sleep(500);
//--- contador del ciclo
   int h_steps=bars/2;
//--- movemos los dos puntos de anclaje horizontalmente a la vez
   for(int i=0;i<h_steps;i++)
     {
       //--- cogemos los siguientes valores
       if(d1<bars-1)
         d1+=1;
       if(d2>1)
         d2-=1;
       //--- movemos los puntos
       if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
       if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
       //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
       if(IsStopped())
         return;
       //--- redibujamos el gráfico
       ChartRedraw();
       // retardo de 0,03 segundo
       Sleep(30);
      }
//--- retardo de 1 segundo
   Sleep(1000);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- eliminamos la línea de tendencia
   TrendDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TRENDBYANGLE

Linea de tendencia por ángulo.



Nota

Para la línea de tendencia se puede indicar el modo de su continuación a la derecha y/o a la izquierda (propiedades OBJPROP_RAY_RIGHT y OBJPROP_RAY_LEFT respectivamente).

Para establecer la inclinación de la línea se puede usar el ángulo, o bien las coordenadas del segundo punto de anclaje.

Ejemplo

El siguiente script crea y desplaza la línea de tendencia en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Línea de tendencia por
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string           InpName="Trend";      // Nombre de la línea
input int              InpDate1=50;          // Fecha del 1-er punto en %
input int              InpPrice1=75;         // Precio del 1-er punto en %
input int              InpAngle=0;           // Ángulo de inclinación de la línea
input color            InpColor=clrRed;      // Color de la línea
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Estilo de la línea
input int             InpWidth=2;          // Grosor de la línea
input bool            InpBack=false;       // Línea al fondo
input bool            InpSelection=true;   // Seleccionar para mover
input bool            InpRayLeft=false;    // Continuación de la línea a la izquierda
input bool            InpRayRight=true;    // Continuación de la línea a la derecha
input bool            InpHidden=true;      // Ocultar en la lista de objetos
input long            InpZOrder=0;         // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea la línea de tendencia por ángulo                            |
//+------------------------------------------------------------------+
bool TrendByAngleCreate(const long            chart_ID=0,        // ID del gráfico
                        const string          name="TrendLine",  // nombre de la línea
                        const int             sub_window=0,      // número de subventa
                        datetime              time=0,            // hora del punto
                        double                price=0,           // precio del punto
                        const double          angle=45.0,        // ángulo de inclinac
                        const color           clr=clrRed,        // color de la línea
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de la línea
                        const int             width=1,           // grosor de la línea
                        const bool            back=false,        // al fondo
                        const bool            selection=true,    // seleccionar para m
                        const bool            ray_left=false,    // continuación de la
                        const bool            ray_right=true,    // continuación de la
                        const bool            hidden=true,       // ocultar en la list
                        const long            z_order=0)         // prioridad para el
  {
//--- para que sea más cómodo mover la línea de tendencia con ratón, crearemos el segu
   datetime time2=0;
   double   price2=0;
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeTrendEmptyPoints(time,price,time2,price2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la línea de tendencia por 2 puntos
   if(!ObjectCreate(chart_ID,name,OBJ_TRENDBYANGLE,sub_window,time,price,time2,price2)
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la línea de tendencia! Código del error = ",GetLastErro
      return(false);
     }
//--- cambiamos el ángulo de inclinación de la línea; durante el cambio del ángulo la
//--- punto de la línea se cambiará automáticamente de acuerdo con los nuevos valores
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- establecemos el color de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de la línea
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la línea con ra
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activamos (true) o desactivamos (false) el modo de continuación de la línea a la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activamos (true) o desactivamos (false) el modo de continuación de la línea a la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia las coordenadas del punto de anclaje de la línea de tendencia
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // ID del gráfico
                      const string name="TrendLine", // nombre de la línea
                      datetime     time=0,           // coordenada del tiempo del punt
                      double       price=0)          // coordenada del precio del punt
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje de la línea de tendencia
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el ángulo de inclinación de la línea de tendencia
//+------------------------------------------------------------------+
bool TrendAngleChange(const long   chart_ID=0,       // ID del gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
                      const string name="TrendLine", // nombre de la línea de tendenci
                      const double angle=45)         // ángulo de inclinación de la lí
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el ángulo de inclinación de la línea de tendencia
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el ángulo de inclinación de la línea de tendencia! Có
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina la línea de tendencia                                    |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // ID del gráfico
                 const string name="TrendLine") // nombre de la línea
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la línea de tendencia
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la línea de tendencia! Código del error = ",GetLastE
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje de la línea y para|
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- si la hora del primer punto no ha sido establecida, se colocará en la barra actu
   if(!time1)
      time1=TimeCurrent();
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- determinamos las coordenadas del segundo punto auxiliar
//--- el segundo punto va a estar 9 barras a la izquierda y tener el mismo precio
   datetime second_point_time[10];
```

© 2000-2025, MetaQuotes Ltd.

```
   CopyTime(Symbol(),Period(),time1,10,second_point_time);
   time2=second_point_time[0];
   price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje de la línea
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError());
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos puntos para trazar la línea
   int d1=InpDate1*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- creamos la línea de tendencia
   if(!TrendByAngleCreate(0,InpName,0,date[d1],price[p1],InpAngle,InpColor,InpStyle,
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover y girar la línea
//--- contador del ciclo
   int v_steps=accuracy/2;
//--- movemos el punto de anclaje y cambiamos el ángulo de inclinación de la línea
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1>1)
         p1-=1;
      //--- movemos el punto
      if(!TrendPointChange(0,InpName,date[d1],price[p1]))
         return;
      if(!TrendAngleChange(0,InpName,18*(i+1)))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos del gráfico
   TrendDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_CYCLES

Líneas cíclicas.



Nota

La distancia entre las líneas se establece a través de las coordenadas del tiempo de dos puntos de anclaje del objeto.

Ejemplo

El siguiente script crea y desplaza las líneas cíclicas el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye las líneas cíclicas en el gráfico."
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="Cycles";   // Nombre del objeto
input int             InpDate1=10;        // Fecha del 1 punto en %
input int             InpPrice1=45;       // Precio del 1 punto en %
input int             InpDate2=20;        // Fecha del 2 punto en %
input int             InpPrice2=55;       // Precio del 2 punto en %
input color           InpColor=clrRed;    // Color de las líneas cíclicas
input ENUM_LINE_STYLE InpStyle=STYLE_DOT; // Estilo de las líneas cíclicas
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpWidth=1;          // Grosor de las líneas cíclicas
input bool             InpBack=false;       // Objeto al fondo
input bool             InpSelection=true;   // Seleccionar para mover
input bool             InpHidden=true;      // Ocultar en la lista de objetos
input long             InpZOrder=0;         // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea las líneas cíclicas                                         |
//+------------------------------------------------------------------+
bool CyclesCreate(const long            chart_ID=0,         // ID del gráfico
                  const string          name="Cycles",      // nombre del objeto
                  const int             sub_window=0,       // número de subventana
                  datetime              time1=0,            // hora del primer punto
                  double                price1=0,           // precio del primer punto
                  datetime              time2=0,            // hora del segundo punto
                  double                price2=0,           // precio del segundo punto
                  const color           clr=clrRed,         // color de las líneas cícl
                  const ENUM_LINE_STYLE style=STYLE_SOLID,  // estilo de las líneas cíc
                  const int             width=1,            // grosor de las líneas cíc
                  const bool            back=false,         // al fondo
                  const bool            selection=true,     // seleccionar para mover
                  const bool            hidden=true,        // ocultar en la lista de c
                  const long            z_order=0)          // prioridad para el clic c
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeCyclesEmptyPoints(time1,price1,time2,price2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos las líneas cíclicas según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_CYCLES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear las líneas cíclicas! Código del error = ",GetLastError
      return(false);
     }
//--- fijamos el color de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- fijamos el estilo de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- fijamos el grosor de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de las líneas con
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool CyclesPointChange(const long   chart_ID=0,    // ID del gráfico
                       const string name="Cycles", // nombre del objeto
                       const int    point_index=0, // número del punto de anclaje
                       datetime     time=0,        // coordenada del tiempo del punto
                       double       price=0)       // coordenada del precio del punto
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina las líneas cíclicas                                      |
//+------------------------------------------------------------------+
bool CyclesDelete(const long   chart_ID=0,    // ID del gráfico
                  const string name="Cycles") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos las líneas cíclicas
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar las líneas cíclicas!! Código del error = ",GetLastEr
      return(false);
     }
//--- ejecución con éxito
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje de las líneas cíclicas y para |
//| los valores vacíos establece los valores por defecto                     |
//+------------------------------------------------------------------+
void ChangeCyclesEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
  {
//--- si la hora del primer punto no ha sido establecida, se colocará en la barra actu
   if(!time1)
      time1=TimeCurrent();
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del segundo punto no ha sido establecida, se colocará a 9 barras a la
   if(!time2)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- colocamos el segundo punto a 9 barras a la izquierda del primero
      time2=temp[0];
     }
//--- si el precio del segundo punto no ha sido establecido, va a coincidir con el pre
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje de las líne
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar las líneas cíclicas
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- creamos la línea de tendencia
   if(!CyclesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje
//--- contador del ciclo
   int h_steps=bars/5;
//--- movemos el segundo punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d2<bars-1)
         d2+=1;
      //--- movemos el punto
      if(!CyclesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   h_steps=bars/4;
//--- movemos el primer punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d1<bars-1)
         d1+=1;
      //--- movemos el punto
      if(!CyclesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el objeto desde el gráfico
   CyclesDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROWED_LINE

Línea con flecha.



Ejemplo

El siguiente script crea y desplaza una línea con flecha en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Línea con flecha\"."
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="ArrowedLine"; // Nombre de la línea
input int             InpDate1=35;           // Fecha del 1 punto en %
input int             InpPrice1=60;          // Precio del 1 punto en %
input int             InpDate2=65;           // Fecha del 2 punto en %
input int             InpPrice2=40;          // Precio del 2 punto en %
input color           InpColor=clrRed;       // Color de la línea
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;   // Estilo de la línea
input int             InpWidth=2;            // Grosor de la línea
input bool            InpBack=false;         // Línea al fondo
input bool            InpSelection=true;     // Seleccionar para mover
input bool            InpHidden=true;        // Ocultar en la lista de objetos
input long            InpZOrder=0;           // Prioridad para el clic del ratón
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Crear la línea con flecha según las coordenadas establecidas
//+------------------------------------------------------------------+
bool ArrowedLineCreate(const long            chart_ID=0,         // ID del gráfico
                       const string          name="ArrowedLine", // nombre de la línea
                       const int             sub_window=0,       // número de subventa
                       datetime              time1=0,            // hora del primer punto
                       double                price1=0,           // precio del primer punto
                       datetime              time2=0,            // hora del segundo punto
                       double                price2=0,           // precio del segundo punto
                       const color           clr=clrRed,         // color de la línea
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // estilo de la línea
                       const int             width=1,            // grosor de la línea
                       const bool            back=false,         // al fondo
                       const bool            selection=true,     // seleccionar para m
                       const bool            hidden=true,        // ocultar en la list
                       const long            z_order=0)          // prioridad para el
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeArrowedLineEmptyPoints(time1,price1,time2,price2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la línea con flecha según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_ARROWED_LINE,sub_window,time1,price1,time2,price
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la línea con flecha! Código del error = ",GetLastError
      return(false);
     }
//--- establecemos el color de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de visualización de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la línea con ra
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje de la línea con flecha                 |
//+------------------------------------------------------------------+
bool ArrowedLinePointChange(const long   chart_ID=0,          // ID del gráfico
                            const string name="ArrowedLine", // nombre de la línea
                            const int    point_index=0,      // número del punto de ar
                            datetime     time=0,             // coordenada del tiempo
                            double       price=0)            // coordenada del precio
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje de la línea
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función elimina la línea con flecha desde el gráfico          |
//+------------------------------------------------------------------+
bool ArrowedLineDelete(const long   chart_ID=0,          // ID del gráfico
                       const string name="ArrowedLine") // nombre de la línea
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la línea con flecha
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la línea con flecha! Código del error = ",GetLastErr
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje de la línea y para|
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeArrowedLineEmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2)
  {
//--- si la hora del primer punto no ha sido establecida, se colocará en la barra actu
   if(!time1)
      time1=TimeCurrent();
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del segundo punto no ha sido establecida, se colocará a 9 barras a la
   if(!time2)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- colocamos el segundo punto a 9 barras a la izquierda del primero
      time2=temp[0];
     }
//--- si el precio del segundo punto no ha sido establecido, va a coincidir con el pre
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje de la línea
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos puntos para trazar la línea
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- creamos la línea con flecha
   if(!ArrowedLineCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje de la línea
//--- contador del ciclo
   int v_steps=accuracy/5;
//--- movemos el segundo punto de anclaje verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p2<accuracy-1)
         p2+=1;
      //--- movemos el punto
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- movemos el primer punto de anclaje verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1>1)
         p1-=1;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- movemos el punto
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de medio segundo
   Sleep(500);
//--- contador del ciclo
   int h_steps=bars/2;
//--- movemos los dos puntos de anclaje horizontalmente a la vez
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos los siguientes valores
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- movemos los puntos
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,03 segundo
      Sleep(30);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos la línea con flecha
   ArrowedLineDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_CHANNEL

Canal equidistante.



Nota

Para el canal equidistante se puede indicar el modo de su continuación a la derecha y/o a la izquierda (propiedades OBJPROP_RAY_RIGHT y OBJPROP_RAY_LEFT respectivamente). Además, se puede establecer el modo de relleno del canal con el color.

Ejemplo

El siguiente script crea y desplaza el canal equidistante en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Canal equidistante\"."
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="Channel";   // Nombre del canal
input int             InpDate1=25;         // Fecha del 1-er punto en %
input int             InpPrice1=60;        // Precio del 1-er punto en %
input int             InpDate2=65;         // Fecha del 2-do punto en %
input int             InpPrice2=80;        // Precio del 2-do punto en %
input int             InpDate3=30;         // Fecha del 3-er punto en %
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpPrice3=40;         // Precio del 3-er punto en %
input color            InpColor=clrRed;      // Color del canal
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH;  // Estilo de las líneas del canal
input int              InpWidth=2;           // Grosor de las líneas del canal
input bool             InpBack=false;        // Canal al fondo
input bool             InpFill=false;        // Relleno del canal con el color
input bool             InpSelection=true;    // Seleccionar para mover
input bool             InpRayLeft=false;     // Continuación del canal a la izquierda
input bool             InpRayRight=false;    // Continuación del canal a la derecha
input bool             InpHidden=true;       // Ocultar en la lista de objetos
input long             InpZOrder=0;          // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el canal equidistante según las coordenadas establecidas    |
//+------------------------------------------------------------------+
bool ChannelCreate(const long            chart_ID=0,        // ID del gráfico
                   const string          name="Channel",    // nombre del canal
                   const int             sub_window=0,      // número de subventana
                   datetime              time1=0,           // hora del primer punto
                   double                price1=0,          // precio del primer punto
                   datetime              time2=0,           // hora del segundo punto
                   double                price2=0,          // precio del segundo punto
                   datetime              time3=0,           // hora del tercer punto
                   double                price3=0,          // precio del tercer punto
                   const color           clr=clrRed,        // color del canal
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de las líneas de
                   const int             width=1,           // grosor de las líneas de
                   const bool            fill=false,        // relleno del canal con e
                   const bool            back=false,        // al fondo
                   const bool            selection=true,    // seleccionar para mover
                   const bool            ray_left=false,    // continuación del canal
                   const bool            ray_right=false,   // continuación del canal
                   const bool            hidden=true,       // ocultar en la lista de
                   const long            z_order=0)         // prioridad para el clic
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el canal según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_CHANNEL,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el canal equidistante! Código del error = ",GetLastErro
      return(false);
     }
//--- fijamos el color del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- establecemos el grosor de las líneas del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activar (true) o desactivar (false) el modo de relleno del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del canal para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activamos (true) o desactivamos (false) el modo de continuación del canal a la i
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activamos (true) o desactivamos (false) el modo de continuación del canal a la d
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje del canal                              |
//+------------------------------------------------------------------+
bool ChannelPointChange(const long   chart_ID=0,     // ID del gráfico
                        const string name="Channel", // nombre del canal
                        const int    point_index=0,  // número del punto de anclaje
                        datetime     time=0,         // coordenada del tiempo del punt
                        double       price=0)        // coordenada del precio del punt
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Elimina el canal                                                 |
//+------------------------------------------------------------------+
bool ChannelDelete(const long   chart_ID=0,     // ID del gráfico
                   const string name="Channel") // nombre del canal
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el canal! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje del canal y para  |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- si la hora del segundo punto (de la derecha) no ha sido establecida, se colocará
   if(!time2)
      time2=TimeCurrent();
//--- si el precio del segundo punto no ha sido establecido, tendrá el valor Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del primer punto (de la izquierda) no ha sido establecida, se colocar
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
//--- si el precio del primer punto no ha sido establecido, lo moveremos a 300 puntos
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si la hora del tercer punto no ha sido establecida, va a coincidir con la hora
   if(!time3)
      time3=time1;
//--- si el precio del tercer punto no ha sido establecido, va a coincidir con el prec
   if(!price3)
      price3=price2;
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del canal
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos puntos para trazar el canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- creamos el canal equidistante
   if(!ChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
```

© 2000-2025, MetaQuotes Ltd.

```
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje del canal
//--- contador del ciclo
   int h_steps=bars/6;
//--- movemos el segundo punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d2<bars-1)
         d2+=1;
      //--- movemos el punto
      if(!ChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- movemos el primer punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d1>1)
         d1-=1;
      //--- movemos el punto
      if(!ChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
```

© 2000-2025, MetaQuotes Ltd.

```
   int v_steps=accuracy/10;
//--- movemos el tercer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p3>1)
         p3-=1;
      //--- movemos el punto
      if(!ChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el canal desde el gráfico
   ChannelDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_STDDEVCHANNEL

Canal de desviación estándar.



Nota

Para el canal de desviación estándar se puede indicar el modo de su continuación a la derecha y/o a la izquierda (propiedades OBJPROP_RAY_RIGHT y OBJPROP_RAY_LEFT respectivamente). Además, se puede establecer el modo de relleno del canal con el color.

Para el cambio del valor de desviación del canal se utiliza la propiedad OBJPROP_DEVIATION.

Ejemplo

El siguiente script crea y desplaza el canal de desviación estándar en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Canal de desviación est
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="StdDevChannel";   // Nombre del canal
input int             InpDate1=10;               // Fecha del 1-er punto en %
input int             InpDate2=40;               // Fecha del 2-do punto en %
input double          InpDeviation=1.0;          // Desviación
input color           InpColor=clrRed;           // Color del canal
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de las líneas del canal
input int             InpWidth=2;                // Grosor de las líneas del canal
input bool            InpFill=false;             // Relleno del canal con el color
input bool            InpBack=false;             // Canal al fondo
input bool            InpSelection=true;         // Seleccionar para mover
input bool            InpRayLeft=false;          // Continuación del canal a la izquie
input bool            InpRayRight=false;         // Continuación del canal a la derech
input bool            InpHidden=true;            // Ocultar en la lista de objetos
input long            InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el canal de desviación estándar según las coordenadas establecidas    |
//+------------------------------------------------------------------+
bool StdDevChannelCreate(const long             chart_ID=0,        // ID del gráfico
                         const string           name="Channel",    // nombre del canal
                         const int              sub_window=0,      // número de subvent
                         datetime               time1=0,           // hora del primer p
                         datetime               time2=0,           // hora del segundo
                         const double           deviation=1.0,     // desviación
                         const color            clr=clrRed,        // color del canal
                         const ENUM_LINE_STYLE  style=STYLE_SOLID, // estilo de las lín
                         const int              width=1,           // grosor de las lín
                         const bool             fill=false,        // relleno del canal
                         const bool             back=false,        // al fondo
                         const bool             selection=true,    // seleccionar para
                         const bool             ray_left=false,    // continuación del
                         const bool             ray_right=false,   // continuación del
                         const bool             hidden=true,       // ocultar en la lis
                         const long             z_order=0)         // prioridad para el
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeChannelEmptyPoints(time1,time2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el canal según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_STDDEVCHANNEL,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el canal de desviación estándar! Código del error = ",G
      return(false);
     }
//--- establecemos la desviación, de ella depende el ancho del canal
   ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation);
//--- fijamos el color del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de las líneas del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activar (true) o desactivar (false) el modo de relleno del canal
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del canal para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activamos (true) o desactivamos (false) el modo de continuación del canal a la i
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activamos (true) o desactivamos (false) el modo de continuación del canal a la d
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje del canal                              |
//+------------------------------------------------------------------+
bool StdDevChannelPointChange(const long   chart_ID=0,      // ID del gráfico
                              const string name="Channel", // nombre del canal
                              const int    point_index=0,  // número del punto de ancl
                              datetime     time=0)         // coordenada del tiempo de
  {
//--- si la hora del punto no está definida, lo movemos a la barra actual
   if(!time)
      time=TimeCurrent();
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia la desviación del canal                                   |
//+------------------------------------------------------------------+
bool StdDevChannelDeviationChange(const long   chart_ID=0,      // ID del gráfico
                                  const string name="Channel", // nombre del canal
                                  const double deviation=1.0)  // desviación
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el ángulo de inclinación de la línea de tendencia
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la desviación del canal! Código del error = ",GetLast
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el canal                                                 |
//+------------------------------------------------------------------+
bool StdDevChannelDelete(const long   chart_ID=0,     // ID del gráfico
                         const string name="Channel") // nombre del canal
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el canal! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje del canal y para  |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,datetime &time2)
  {
//--- si la hora del segundo punto no ha sido establecida, se colocará en la barra act
   if(!time2)
      time2=TimeCurrent();
//--- si la hora del primer punto no ha sido establecida, se colocará a 9 barras a la
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el segundo punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del canal
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos puntos para trazar el canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- creamos el canal de desviación estándar
   if(!StdDevChannelCreate(0,InpName,0,date[d1],date[d2],InpDeviation,InpColor,InpStyl
      InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el canal por la horizontal a la derecha y ampliarlo
//--- contador del ciclo
   int h_steps=bars/2;
//--- movemos el canal
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos los siguientes valores
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- movemos los puntos de anclaje
      if(!StdDevChannelPointChange(0,InpName,0,date[d1]))
         return;
      if(!StdDevChannelPointChange(0,InpName,1,date[d2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   double v_steps=InpDeviation*2;
//--- ampliamos el canal
   for(double i=InpDeviation;i<v_steps;i+=10.0/accuracy)
     {
      if(!StdDevChannelDeviationChange(0,InpName,i))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el canal desde el gráfico
   StdDevChannelDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

OBJ_REGRESSION

Canal de regresión lineal.



Nota

Para el canal de regresión lineal se puede indicar el modo de su continuación a la derecha y/o a la izquierda (propiedades OBJPROP_RAY_RIGHT y OBJPROP_RAY_LEFT respectivamente). Además, se puede establecer el modo de relleno del canal con el color.

Ejemplo

El siguiente script crea y desplaza el canal de regresión lineal en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Canal de regresión line
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="Regression"; // Nombre del canal
input int             InpDate1=10;          // Fecha del 1-er punto en %
input int             InpDate2=40;          // Fecha del 2-do punto en %
input color           InpColor=clrRed;      // Color del canal
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Estilo de las líneas del canal
input int             InpWidth=2;           // Grosor de las líneas del canal
```

© 2000-2025, MetaQuotes Ltd.

```
input bool             InpFill=false;        // Relleno del canal con el color
input bool             InpBack=false;        // Canal al fondo
input bool             InpSelection=true;    // Seleccionar para mover
input bool             InpRayLeft=false;     // Continuación del canal a la izquierda
input bool             InpRayRight=false;    // Continuación del canal a la derecha
input bool             InpHidden=true;       // Ocultar en la lista de objetos
input long             InpZOrder=0;          // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el canal regresión lineal según las coordenadas establecidas      |
//+------------------------------------------------------------------+
bool RegressionCreate(const long             chart_ID=0,        // ID del gráfico
                      const string           name="Regression", // nombre del canal
                      const int              sub_window=0,      // número de subventana
                      datetime               time1=0,           // hora del primer punt
                      datetime               time2=0,           // hora del segundo pun
                      const color            clr=clrRed,        // color del canal
                      const ENUM_LINE_STYLE  style=STYLE_SOLID, // estilo de las líneas
                      const int              width=1,           // grosor de las líneas
                      const bool             fill=false,        // relleno del canal co
                      const bool             back=false,        // al fondo
                      const bool             selection=true,    // seleccionar para mov
                      const bool             ray_left=false,    // continuación del car
                      const bool             ray_right=false,   // continuación del car
                      const bool             hidden=true,       // ocultar en la lista
                      const long             z_order=0)         // prioridad para el cl
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeRegressionEmptyPoints(time1,time2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el canal según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_REGRESSION,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el canal de regresión lineal! Código del error = ",GetL
      return(false);
     }
//--- fijamos el color del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de las líneas del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activar (true) o desactivar (false) el modo de relleno del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del canal para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
```

© 2000-2025, MetaQuotes Ltd.

```
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activamos (true) o desactivamos (false) el modo de continuación del canal a la i
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activamos (true) o desactivamos (false) el modo de continuación del canal a la d
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje del canal                              |
//+------------------------------------------------------------------+
bool RegressionPointChange(const long   chart_ID=0,     // ID del gráfico
                           const string name="Channel", // nombre del canal
                           const int    point_index=0,  // número del punto de anclaje
                           datetime     time=0)         // coordenada del tiempo del p
  {
//--- si la hora del punto no está definida, lo movemos a la barra actual
   if(!time)
      time=TimeCurrent();
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el canal                                                 |
//+------------------------------------------------------------------+
bool RegressionDelete(const long   chart_ID=0,     // ID del gráfico
                      const string name="Channel") // nombre del canal
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el canal
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el canal! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje del canal y para  |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeRegressionEmptyPoints(datetime &time1,datetime &time2)
  {
//--- si la hora del segundo punto no ha sido establecida, se colocará en la barra act
   if(!time2)
      time2=TimeCurrent();
//--- si la hora del primer punto no ha sido establecida, se colocará a 9 barras a la
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del canal
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos puntos para trazar el canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- creamos el canal de regresión lineal
   if(!RegressionCreate(0,InpName,0,date[d1],date[d2],InpColor,InpStyle,InpWidth,
      InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el canal por la horizontal a la derecha
//--- contador del ciclo
   int h_steps=bars/2;
//--- movemos el canal
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos los siguientes valores
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- movemos los puntos de anclaje
      if(!RegressionPointChange(0,InpName,0,date[d1]))
         return;
      if(!RegressionPointChange(0,InpName,1,date[d2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
```

© 2000-2025, MetaQuotes Ltd.

```
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el canal desde el gráfico
   RegressionDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_PITCHFORK

Tridente Andrews.



Nota

Para el "Tridente Andrews" se puede indicar el modo de su continuación a la derecha y/o a la izquierda (propiedades OBJPROP_RAY_RIGHT y OBJPROP_RAY_LEFT respectivamente).

Además, se puede indicar el número de líneas-niveles, su valor y el color.

Ejemplo

El siguiente script crea y desplaza el "Tridente Andrews" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Tridente Andrews\"."
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="Pitchfork";  // Nombre del tridente
input int             InpDate1=14;          // Fecha del 1-er punto en %
input int             InpPrice1=40;         // Precio del 1-er punto en %
input int             InpDate2=18;          // Fecha del 2-do punto en %
input int             InpPrice2=50;         // Precio del 2-do punto en %
input int             InpDate3=18;          // Fecha del 3-er punto en %
input int             InpPrice3=30;         // Precio del 3-er punto en %
```

© 2000-2025, MetaQuotes Ltd.

```
input color           InpColor=clrRed;      // Color del tridente
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Estilo de las líneas del tridente
input int             InpWidth=1;           // Grosor de las líneas del tridente
input bool            InpBack=false;        // Tridente al fondo
input bool            InpSelection=true;    // Seleccionar para mover
input bool            InpRayLeft=false;     // Continuación del tridente a la izquierd
input bool            InpRayRight=false;    // Continuación del tridente a la derecha
input bool            InpHidden=true;       // Ocultar en la lista de objetos
input long            InpZOrder=0;          // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el "Tridente Andrews" según las coordenadas establecidas    |
//+------------------------------------------------------------------+
bool PitchforkCreate(const long            chart_ID=0,        // ID del gráfico
                     const string          name="Pitchfork",  // nombre del tridente
                     const int             sub_window=0,      // número de subventana
                     datetime              time1=0,           // hora del primer punto
                     double                price1=0,          // precio del primer pun
                     datetime              time2=0,           // hora del segundo punt
                     double                price2=0,          // precio del segundo pu
                     datetime              time3=0,           // hora del tercer punto
                     double                price3=0,          // precio del tercer pun
                     const color           clr=clrRed,        // color de las líneas d
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de las líneas
                     const int             width=1,           // grosor de las líneas
                     const bool            back=false,        // al fondo
                     const bool            selection=true,    // seleccionar para move
                     const bool            ray_left=false,    // continuación del tric
                     const bool            ray_right=false,   // continuación del tric
                     const bool            hidden=true,       // ocultar en la lista d
                     const long            z_order=0)         // prioridad para el cli
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el "Tridente Andrews" según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_PITCHFORK,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": ¡Falo al crear el \"Tridente Andrews\"! Código del error = ",GetLastErr
      return(false);
     }
//--- establecemos el color
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- fijamos el grosor de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del tridente para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activamos (true) o desactivamos (false) el modo de continuación del tridente a l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activamos (true) o desactivamos (false) el modo de continuación del tridente a l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Establece el número de los niveles del "Tridente Andrews" y sus parámetros
//+------------------------------------------------------------------+
bool PitchforkLevelsSet(int             levels,           // número de las líneas del
                        double          &values[],        // valores de las líneas del
                        color           &colors[],        // color de las líneas del n
                        ENUM_LINE_STYLE &styles[],        // estilo de las líneas del
                        int             &widths[],        // grosor de las líneas del
                        const long      chart_ID=0,       // ID del gráfico
                        const string    name="Pitchfork") // nombre del tridente
  {
//--- comprobamos los tamaños de los arrays
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": ¡Error. La longitud del array no corresponde al número de
      return(false);
     }
//--- establecemos el número de los niveles
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- establecemos las propiedades de los niveles en el ciclo
   for(int i=0;i<levels;i++)
     {
      //--- valor del nivel
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- color del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- grosor del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- descripción del nivel
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje del "Tridente Andrews"                 |
//+------------------------------------------------------------------+
bool PitchforkPointChange(const long   chart_ID=0,       // ID del gráfico
                          const string name="Pitchfork", // nombre del canal
                          const int    point_index=0,    // número del punto de anclaj
                          datetime     time=0,           // coordenada del tiempo del
                          double       price=0)          // coordenada del precio del
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el "Tridente Andrews"                                    |
//+------------------------------------------------------------------+
bool PitchforkDelete(const long   chart_ID=0,       // ID del gráfico
                     const string name="Pitchfork") // nombre del canal
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el \"Tridente Andrews\"! Código del error = ",GetLas
      return(false);
     }
//--- ejecución con éxito
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje del "Tridente Andrews" y para
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- si la hora del segundo punto (el superior derecho) no ha sido establecida, se co
   if(!time2)
      time2=TimeCurrent();
//--- si el precio del segundo punto no ha sido establecido, tendrá el valor Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del primer punto (de la izquierda) no ha sido establecida, se colocar
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
//--- si el precio del primer punto no ha sido establecido, lo moveremos a 200 puntos
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si la hora del tercer punto no ha sido establecida, va a coincidir con la hora d
   if(!time3)
      time3=time2;
//--- si el precio del tercer punto no ha sido establecido, lo moveremos a 200 puntos
   if(!price3)
      price3=price1-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
```

© 2000-2025, MetaQuotes Ltd.

```
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del "Trider
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el "Tridente Andrewa"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- creamos el Tridente
   if(!PitchforkCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pric
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje del tridente
//--- contador del ciclo
   int v_steps=accuracy/10;
//--- movemos el primer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1>1)
         p1-=1;
      //--- movemos el punto
```

© 2000-2025, MetaQuotes Ltd.

```
      if(!PitchforkPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   int h_steps=bars/8;
//--- movemos el tercer punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d3<bars-1)
         d3+=1;
      //--- movemos el punto
      if(!PitchforkPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   v_steps=accuracy/10;
//--- movemos el segundo punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p2>1)
         p2-=1;
      //--- movemos el punto
      if(!PitchforkPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el tridente desde el gráfico
   PitchforkDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNLINE

Línea de Gann.



Nota

Para la "Línea de Gann" se puede indicar el modo de su continuación a la derecha y/o a la izquierda (propiedades OBJPROP_RAY_RIGHT y OBJPROP_RAY_LEFT respectivamente).

Para establecer la inclinación de la línea se puede usar el ángulo de Gann con la escala, o bien las coordenadas del segundo punto de anclaje.

Ejemplo

El siguiente script crea y desplaza la "Línea de Gann" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Línea de Gann\"."
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="GannLine";        // Nombre de la línea
input int             InpDate1=20;               // Fecha del 1-er punto en %
input int             InpPrice1=75;              // Precio del 1-er punto en %
input int             InpDate2=80;               // Fecha del 2-do punto en %
input double          InpAngle=0.0;              // Ángulo de Gann
```

© 2000-2025, MetaQuotes Ltd.

```
input double           InpScale=1.0;              // Escala
input color            InpColor=clrRed;           // Color de la línea
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Estilo de la línea
input int              InpWidth=2;                // Grosor de la línea
input bool             InpBack=false;             // Línea al fondo
input bool             InpSelection=true;         // Seleccionar para mover
input bool             InpRayLeft=false;          // Continuación de la línea a la izqu
input bool             InpRayRight=true;          // Continuación de la línea a la dere
input bool             InpHidden=true;            // Ocultar en la lista de objetos
input long             InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea la "Línea de Gann" según las coordenadas, el ángulo y la escala            |
//+------------------------------------------------------------------+
bool GannLineCreate(const long             chart_ID=0,        // ID del gráfico
                    const string           name="GannLine",   // nombre de la línea
                    const int              sub_window=0,      // número de subventana
                    datetime               time1=0,           // hora del primer punto
                    double                 price1=0,          // precio del primer punt
                    datetime               time2=0,           // hora del segundo punto
                    const double           angle=1.0,         // ángulo de Gann
                    const double           scale=1.0,         // escala
                    const color            clr=clrRed,        // color de la línea
                    const ENUM_LINE_STYLE  style=STYLE_SOLID, // estilo de la línea
                    const int              width=1,           // grosor de la línea
                    const bool             back=false,        // al fondo
                    const bool             selection=true,    // seleccionar para mover
                    const bool             ray_left=false,    // continuación de la lín
                    const bool             ray_right=true,    // continuación de la lín
                    const bool             hidden=true,       // ocultar en la lista de
                    const long             z_order=0)         // prioridad para el clic
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeGannLineEmptyPoints(time1,price1,time2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la "Línea de Gann" según las coordenadas establecidas
//--- la coordenada correcta del precio del segundo punto de anclaje será redefinida
//--- automáticamente después del cambio del ángulo de Gann y (o) la escala,
   if(!ObjectCreate(chart_ID,name,OBJ_GANNLINE,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la \"Línea de Gann\"! Código del error = ",GetLastError
      return(false);
     }
//--- cambiamos el ángulo de Gann
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- cambiamos la escala (número de pips por barra)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- establecemos el color de la línea
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de visualización de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección de la línea para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activamos (true) o desactivamos (false) el modo de continuación de la línea a la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activamos (true) o desactivamos (false) el modo de continuación de la línea a la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje de la "Línea de Gann"                  |
//+------------------------------------------------------------------+
bool GannLinePointChange(const long   chart_ID=0,      // ID del gráfico
                         const string name="GannLine", // nombre de la línea
                         const int    point_index=0,   // número del punto de anclaje
                         datetime     time=0,          // coordenada del tiempo del pu
                         double       price=0)         // coordenada del precio del pu
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje de la línea
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Cambia el ángulo de Gann                                         |
//+------------------------------------------------------------------+
bool GannLineAngleChange(const long   chart_ID=0,      // ID del gráfico
                         const string name="GannLine", // nombre de la línea
                         const double angle=1.0)       // ángulo de Gann
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el ángulo de Gann
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el ángulo de Gann! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia la escala de la "Línea de Gann"                           |
//+------------------------------------------------------------------+
bool GannLineScaleChange(const long   chart_ID=0,      // ID del gráfico
                         const string name="GannLine", // nombre de la línea
                         const double scale=1.0)       // escala
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos la escala (número de pips por barra)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la escala! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función elimina la "Línea de Gann" desde el gráfico           |
//+------------------------------------------------------------------+
bool GannLineDelete(const long   chart_ID=0,      // ID del gráfico
                    const string name="GannLine") // nombre de la línea
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la línea de Gann
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": ¡Fallo al eliminar la \"Línea de Gann\"! Código del error = ",GetLastEr
       return(false);
      }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje de la línea de Gann y para      |
//| los valores vacíos establece los valores por defecto                           |
//+------------------------------------------------------------------+
void ChangeGannLineEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- si la hora del segundo punto no ha sido establecida, se colocará en la barra act
   if(!time2)
      time2=TimeCurrent();
//--- si la hora del primer punto no ha sido establecida, se colocará a 9 barras a la
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje de la línea
   datetime date[];
```

© 2000-2025, MetaQuotes Ltd.

```
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar la línea de Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- creamos la línea de Gann
   if(!GannLineCreate(0,InpName,0,date[d1],price[p1],date[d2],InpAngle,InpScale,InpCol
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el punto de anclaje de la línea y cambiar el ángulo
//--- contador del ciclo
   int v_steps=accuracy/2;
//--- movemos el primer punto de anclaje verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1>1)
         p1-=1;
      //--- movemos el punto
      if(!GannLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- retardo de medio segundo
   Sleep(500);
//--- determinamos el valor actual del ángulo de Gann (se ha cambiado
//--- tras el desplazamiento del primer punto de anclaje)
   double curr_angle;
   if(!ObjectGetDouble(0,InpName,OBJPROP_ANGLE,0,curr_angle))
      return;
//--- contador del ciclo
   v_steps=accuracy/8;
//--- cambiamos el ángulo de Gann
   for(int i=0;i<v_steps;i++)
     {
      if(!GannLineAngleChange(0,InpName,curr_angle-0.05*i))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos la línea desde el gráfico
   GannLineDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNFAN

Abanico de Gann.



Nota

Para el "Abanico de Gann" se puede indicar el tipo del trend usando la enumeración ENUM_GANN_DIRECTION. Regulando los valores de la escala (OBJPROP_SCALE), se puede cambiar el ángulo de inclinación de las líneas del abanico.

Ejemplo

El siguiente script crea y desplaza el "Abanico de Gann" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Abanico de Gann\"."
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="GannFan";         // Nombre del abanico
input int             InpDate1=15;               // Fecha del 1-er punto en %
input int             InpPrice1=25;              // Precio del 1-er punto en %
input int             InpDate2=85;               // Fecha del 2-do punto en %
input double          InpScale=2.0;              // Escala
input bool            InpDirection=false;        // Dirección del trend
```

© 2000-2025, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // Color del abanico
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de las líneas del abanico
input int             InpWidth=1;                // Grosor de las líneas del abanico
input bool            InpBack=false;             // Abanico al fondo
input bool            InpSelection=true;         // Seleccionar para mover
input bool            InpHidden=true;            // Ocultar en la lista de objetos
input long            InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el "Abanico de Gann"                                        |
//+------------------------------------------------------------------+
bool GannFanCreate(const long            chart_ID=0,        // ID del gráfico
                   const string          name="GannFan",    // nombre del abanico
                   const int             sub_window=0,      // número de subventana
                   datetime              time1=0,           // hora del primer punto
                   double                price1=0,          // precio del primer punto
                   datetime              time2=0,           // hora del segundo punto
                   const double          scale=1.0,         // escala
                   const bool            direction=true,    // dirección del trend
                   const color           clr=clrRed,        // color del abanico
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de las líneas de
                   const int             width=1,           // grosor de las líneas de
                   const bool            back=false,        // al fondo
                   const bool            selection=true,    // seleccionar para mover
                   const bool            hidden=true,       // ocultar en la lista de
                   const long            z_order=0)         // prioridad para el clic
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeGannFanEmptyPoints(time1,price1,time2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el "Abanico de Gann" según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_GANNFAN,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el \"Abanico de Gann\"! Código del error = ",GetLastErr
      return(false);
     }
//--- cambiamos la escala (número de pips por barra)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- cambiamos la dirección del trend del "Abanico de Gann" (true - hacia abajo, fals
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- establecemos el color del abanico
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas del abanico
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de las líneas del abanico
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- activar (true) o desactivar (false) el modo de selección del abanico para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje del "Abanico de Gann"                  |
//+------------------------------------------------------------------+
bool GannFanPointChange(const long   chart_ID=0,     // ID del gráfico
                        const string name="GannFan", // nombre del abanico
                        const int    point_index=0,  // número del punto de anclaje
                        datetime     time=0,         // coordenada del tiempo del punt
                        double       price=0)        // coordenada del precio del punt
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje del abanico
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia la escala del "Abanico de Gann"                           |
//+------------------------------------------------------------------+
bool GannFanScaleChange(const long   chart_ID=0,     // ID del gráfico
                        const string name="GannFan", // nombre del abanico
                        const double scale=1.0)      // escala
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos la escala (número de pips por barra)
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la escala! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia la dirección del trend del "Abanico de Gann"
//+------------------------------------------------------------------+
bool GannFanDirectionChange(const long   chart_ID=0,     // ID del gráfico
                            const string name="GannFan", // nombre del abanico
                            const bool   direction=true) // dirección del trend
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos la dirección del trend del "Abanico de Gann"
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la dirección del trend! Código del error = ",GetLastE
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función elimina el "Abanico de Gann" desde el gráfico
//+------------------------------------------------------------------+
bool GannFanDelete(const long   chart_ID=0,     // ID del gráfico
                   const string name="GannFan") // nombre del abanico
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el Abanico de Gann
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el \"Abanico de Gann\"! Código del error = ",GetLast
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje del Abanico de Gann y para      |
//| los valores vacíos establece los valores por defecto                           |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeGannFanEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- si la hora del segundo punto no ha sido establecida, se colocará en la barra act
   if(!time2)
      time2=TimeCurrent();
//--- si la hora del primer punto no ha sido establecida, se colocará a 9 barras a la
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del abanico
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el abanico de Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- creamos el Abanico de Gann
   if(!GannFanCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el punto de anclaje del abanico
//--- contador del ciclo
   int v_steps=accuracy/2;
//--- movemos el primer punto de anclaje verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1<accuracy-1)
         p1+=1;
      //--- movemos el punto
      if(!GannFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- cambiamos la dirección del trend del abanico hacia abajo
   GannFanDirectionChange(0,InpName,true);
//--- redibujamos el gráfico
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el abanico desde el gráfico
   GannFanDelete(0,InpName);
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNGRID

Retícula de Gann.



Nota

Para el "Retícula de Gann" se puede indicar el tipo del trend usando la enumeración ENUM_GANN_DIRECTION. Regulando los valores de la escala (OBJPROP_SCALE), se puede cambiar el ángulo de inclinación de las líneas de la retícula.

Ejemplo

El siguiente script crea y desplaza la "Retícula de Gann" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Retícula de Gann\"."
#property description "Las coordenadas de los puntos de anclaje de la retícula se esta
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="GannGrid";        // Nombre de la retícula
input int             InpDate1=15;               // Fecha del 1-er punto en %
input int             InpPrice1=25;              // Precio del 1-er punto en %
input int             InpDate2=35;               // Fecha del 2-do punto en %
input double          InpScale=3.0;              // Escala
input bool            InpDirection=false;        // Dirección del trend
```

© 2000-2025, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // Color de la retícula
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de las líneas de la retícul
input int             InpWidth=1;                // Grosor de las líneas del abanico
input bool            InpBack=false;             // Retícula al fondo
input bool            InpSelection=true;         // Seleccionar para mover
input bool            InpHidden=true;            // Ocultar en la lista de objetos
input long            InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea la "Retícula de Gann"                                       |
//+------------------------------------------------------------------+
bool GannGridCreate(const long            chart_ID=0,        // ID del gráfico
                    const string          name="GannGrid",   // nombre de la retícula
                    const int             sub_window=0,      // número de subventana
                    datetime              time1=0,           // hora del primer punto
                    double                price1=0,          // precio del primer punt
                    datetime              time2=0,           // hora del segundo punto
                    const double          scale=1.0,         // escala
                    const bool            direction=true,    // dirección del trend
                    const color           clr=clrRed,        // color de la retícula
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de las líneas
                    const int             width=1,           // grosor de las líneas
                    const bool            back=false,        // al fondo
                    const bool            selection=true,    // seleccionar para mover
                    const bool            hidden=true,       // ocultar en la lista de
                    const long            z_order=0)         // prioridad para el clic
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeGannGridEmptyPoints(time1,price1,time2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la "Retícula de Gann" según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_GANNGRID,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la \"Retícula de Gann\"! Código del error = ",GetLastEr
      return(false);
     }
//--- cambiamos la escala (número de pips por barra)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- cambiamos la dirección del trend de la "Retícula de Gann" (true - hacia abajo,
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- establecemos el color de la retícula
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas de la retícula
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de las líneas de la retícula
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- activar (true) o desactivar (false) el modo de selección de la retícula para mov
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje de la "Retícula de Gann"               |
//+------------------------------------------------------------------+
bool GannGridPointChange(const long   chart_ID=0,      // ID del gráfico
                         const string name="GannGrid", // nombre de la retícula
                         const int    point_index=0,   // número del punto de anclaje
                         datetime     time=0,          // coordenada del tiempo del pu
                         double       price=0)         // coordenada del precio del pu
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje de la retícula
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia la escala de la "Retícula de Gann"                        |
//+------------------------------------------------------------------+
bool GannGridScaleChange(const long   chart_ID=0,      // ID del gráfico
                         const string name="GannGrid", // nombre de la retícula
                         const double scale=1.0)       // escala
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos la escala (número de pips por barra)
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la escala! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia la dirección del trend de la "Retícula de Gann"
//+------------------------------------------------------------------+
bool GannGridDirectionChange(const long   chart_ID=0,      // ID del gráfico
                             const string name="GannGrid", // nombre de la retícula
                             const bool   direction=true)  // dirección del trend
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos la dirección del trend de la "Retícula de Gann"
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la dirección del trend! Código del error = ",GetLastE
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función elimina la "Retícula de Gann" desde el gráfico
//+------------------------------------------------------------------+
bool GannGridDelete(const long   chart_ID=0,      // ID del gráfico
                    const string name="GannGrid") // nombre de la retícula
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la Retícula de Gann
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la \"Retícula de Gann\"! Código del error = ",GetLas
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje de la Retícula de Gann y para
//| los valores vacíos establece los valores por defecto                     |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeGannGridEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- si la hora del segundo punto no ha sido establecida, se colocará en la barra act
   if(!time2)
      time2=TimeCurrent();
//--- si la hora del primer punto no ha sido establecida, se colocará a 9 barras a la
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje de la retíc
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar la Retícula de Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- creamos la Retícula de Gann
   if(!GannGridCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje de la retícula
//--- contador del ciclo
   int v_steps=accuracy/4;
//--- movemos el primer punto de anclaje verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1<accuracy-1)
         p1+=1;
      if(!GannGridPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   int h_steps=bars/4;
//--- movemos el segundo punto de anclaje por la horizontal
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d2<bars-1)
         d2+=1;
      if(!GannGridPointChange(0,InpName,1,date[d2],0))
```

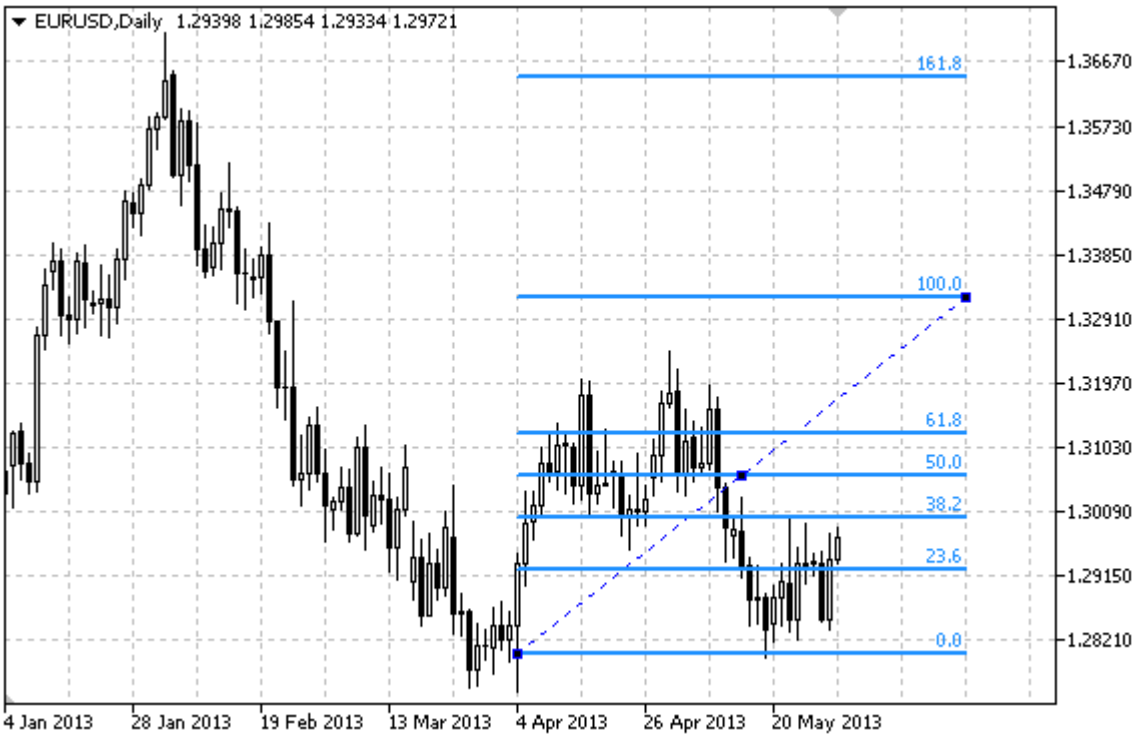
© 2000-2025, MetaQuotes Ltd.

```
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- cambiamos la dirección del trend de la retícula hacia abajo
   GannGridDirectionChange(0,InpName,true);
//--- redibujamos el gráfico
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos la retícula desde el gráfico
   GannGridDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBO

Retrocesos de Fibonacci.



Nota

Para los "Retrocesos de Fibonacci" se puede indicar el modo de su continuación a la derecha y/o a la izquierda (propiedades OBJPROP_RAY_RIGHT y OBJPROP_RAY_LEFT respectivamente).

Además, se puede indicar el número de líneas-niveles, su valor y el color.

Ejemplo

El siguiente script crea y desplaza los "Retrocesos de Fibonacci" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Retrocesos de Fibonacci
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="FiboLevels";      // Nombre del objeto
input int             InpDate1=10;               // Fecha del 1-er punto en %
input int             InpPrice1=65;              // Precio del 1-er punto en %
input int             InpDate2=90;               // Fecha del 2-do punto en %
input int             InpPrice2=85;              // Precio del 2-do punto en %
input color           InpColor=clrRed;           // Color del objeto
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de la línea
input int             InpWidth=2;                // Grosor de la línea
input bool            InpBack=false;             // Objeto al fondo
input bool            InpSelection=true;         // Seleccionar para mover
input bool            InpRayLeft=false;          // Continuación del objeto a la izqui
input bool            InpRayRight=false;         // Continuación del objeto a la derec
input bool            InpHidden=true;            // Ocultar en la lista de objetos
input long            InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea los "Retrocesos de Fibonacci" según las coordenadas establecidas
//+------------------------------------------------------------------+
bool FiboLevelsCreate(const long            chart_ID=0,        // ID del gráfico
                      const string          name="FiboLevels", // nombre del objeto
                      const int             sub_window=0,      // número de subventana
                      datetime              time1=0,           // hora del primer punt
                      double                price1=0,          // precio del primer pu
                      datetime              time2=0,           // hora del segundo pun
                      double                price2=0,          // precio del segundo p
                      const color           clr=clrRed,        // color del objeto
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de las líneas
                      const int             width=1,           // grosor de las líneas
                      const bool            back=false,        // al fondo
                      const bool            selection=true,    // seleccionar para mov
                      const bool            ray_left=false,    // continuación del obj
                      const bool            ray_right=false,   // continuación del obj
                      const bool            hidden=true,       // ocultar en la lista
                      const long            z_order=0)         // prioridad para el cl
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeFiboLevelsEmptyPoints(time1,price1,time2,price2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos los "Retrocesos de Fibonacci" según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_FIBO,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": ¡Falo al crear los \"Retrocesos de Fibonacci\"! Código del error = ",Ge
      return(false);
     }
//--- establecemos el color
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del objeto para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
```

© 2000-2025, MetaQuotes Ltd.

```
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activamos (true) o desactivamos (false) el modo de continuación del objeto a la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activamos (true) o desactivamos (false) el modo de continuación del objeto a la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear con el ratón sobre
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Establece el número de los niveles y sus parámetros              |
//+------------------------------------------------------------------+
bool FiboLevelsSet(int              levels,           // número de las líneas del nive
                   double           &values[],        // valores de las líneas del niv
                   color            &colors[],        // color de las líneas del nivel
                   ENUM_LINE_STYLE &styles[],         // estilo de las líneas del nive
                   int              &widths[],        // grosor de las líneas del nive
                   const long       chart_ID=0,       // ID del gráfico
                   const string     name="FiboLevels") // nombre del objeto
  {
//--- comprobamos los tamaños de los arrays
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) || levels!=ArraySize(widt
     {
      Print(__FUNCTION__,": ¡Error. La longitud del array no corresponde al número de
      return(false);
     }
//--- establecemos el número de los niveles
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- establecemos las propiedades de los niveles en el ciclo
   for(int i=0;i<levels;i++)
     {
      //--- valor del nivel
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- color del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- grosor del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descripción del nivel
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- ejecución con éxito
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje de los "Retrocesos de Fibonacci"       |
//+------------------------------------------------------------------+
bool FiboLevelsPointChange(const long   chart_ID=0,        // ID del gráfico
                           const string name="FiboLevels", // nombre del objeto
                           const int    point_index=0,     // número del punto de ancl
                           datetime     time=0,            // coordenada del tiempo de
                           double       price=0)           // coordenada del precio de
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina los "Retrocesos de Fibonacci"                            |
//+------------------------------------------------------------------+
bool FiboLevelsDelete(const long   chart_ID=0,        // ID del gráfico
                      const string name="FiboLevels") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Falo al eliminar los \"Retrocesos de Fibonacci\"! Código del error = "
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| //| Comprueba los valores de los puntos de anclaje de los "Retrocesos de Fibonacci
//| los valores vacíos establece los valores por defecto             |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeFiboLevelsEmptyPoints(datetime &time1,double &price1,
                                 datetime &time2,double &price2)
  {
//--- si la hora del segundo punto no ha sido establecida, se colocará en la barra act
   if(!time2)
      time2=TimeCurrent();
//--- si el precio del segundo punto no ha sido establecido, tendrá el valor Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del primer punto no ha sido establecida, se colocará a 9 barras a la
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
//--- si el precio del primer punto no ha sido establecido, lo moveremos a 200 puntos
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje de los "Ret
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
       return;
      }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar los "Retrocesos de Fibonacci"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- creamos el objeto
   if(!FiboLevelsCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
      {
       return;
      }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje
//--- contador del ciclo
   int v_steps=accuracy*2/5;
//--- movemos el primer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1>1)
         p1-=1;
      //--- movemos el punto
      if(!FiboLevelsPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   v_steps=accuracy*4/5;
//--- movemos el segundo punto de anclaje
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p2>1)
         p2-=1;
      //--- movemos el punto
      if(!FiboLevelsPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el objeto desde el gráfico
   FiboLevelsDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOTIMES

Zonas temporales de Fibonacci.



Nota

Para las "Zonas temporales de Fibonacci" se puede indicar el número de sus líneas-niveles, su valor y el color.

Ejemplo

El siguiente script crea y desplaza las "Zonas temporales de Fibonacci" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Zonas temporales de Fib
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="FiboTimes";       // Nombre del objeto
input int             InpDate1=10;               // Fecha del 1-er punto en %
input int             InpPrice1=45;              // Precio del 1-er punto en %
input int             InpDate2=20;               // Fecha del 2-do punto en %
input int             InpPrice2=55;              // Precio del 2-do punto en %
input color           InpColor=clrRed;           // Color del objeto
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de la línea
input int             InpWidth=2;                // Grosor de la línea
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpBack=false;             // Objeto al fondo
input bool            InpSelection=true;         // Seleccionar para mover
input bool            InpHidden=true;            // Ocultar en la lista de objetos
input long            InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea las "Zonas temporales de Fibonacci" según las coordinadas establecidas
//+------------------------------------------------------------------+
bool FiboTimesCreate(const long            chart_ID=0,        // ID del gráfico
                     const string          name="FiboTimes",  // nombre del objeto
                     const int             sub_window=0,      // número de subventana
                     datetime              time1=0,           // hora del primer punto
                     double                price1=0,          // precio del primer pun
                     datetime              time2=0,           // hora del segundo punt
                     double                price2=0,          // precio del segundo pu
                     const color           clr=clrRed,        // color del objeto
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de las líneas
                      const int             width=1,           // grosor de las líneas
                     const bool            back=false,        // al fondo
                     const bool            selection=true,    // seleccionar para move
                     const bool            hidden=true,       // ocultar en la lista d
                     const long            z_order=0)         // prioridad para el cli
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeFiboTimesEmptyPoints(time1,price1,time2,price2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos las "Zonas temporales de Fibonacci" según las coordinadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOTIMES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": ¡Falo al crear las \"Zonas temporales de Fibonacci\"! Código del error
      return(false);
     }
//--- establecemos el color
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del objeto para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Establece el número de los niveles y sus parámetros              |
//+------------------------------------------------------------------+
bool FiboTimesLevelsSet(int              levels,           // número de las líneas del
                        double           &values[],        // valores de las líneas del
                        color            &colors[],        // color de las líneas del r
                        ENUM_LINE_STYLE &styles[],         // estilo de las líneas del
                        int              &widths[],        // grosor de las líneas del
                        const long       chart_ID=0,       // ID del gráfico
                        const string     name="FiboTimes") // nombre del objeto
  {
//--- comprobamos los tamaños de los arrays
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": ¡Error. La longitud del array no corresponde al número de
      return(false);
     }
//--- establecemos el número de los niveles
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- establecemos las propiedades de los niveles en el ciclo
   for(int i=0;i<levels;i++)
     {
      //--- valor del nivel
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- color del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- grosor del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descripción del nivel
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(values[i],1));
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje de las "Zonas temporales de Fibonacci"
//+------------------------------------------------------------------+
bool FiboTimesPointChange(const long   chart_ID=0,       // ID del gráfico
                          const string name="FiboTimes", // nombre del objeto
                          const int    point_index=0,    // número del punto de ancla
                          datetime     time=0,           // coordenada del tiempo del
```

© 2000-2025, MetaQuotes Ltd.

```
                           double       price=0)          // coordenada del precio del
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina las "Zonas temporales de Fibonacci"                      |
//+------------------------------------------------------------------+
bool FiboTimesDelete(const long   chart_ID=0,       // ID del gráfico
                     const string name="FiboTimes") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Falo al eliminar las \"Zonas temporales de Fibonacci\"! Código del err
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| //| Comprueba los valores de los puntos de anclaje de las "Zonas temporales de Fib
//| para los valores vacíos establece los valores por defecto        |
//+------------------------------------------------------------------+
void ChangeFiboTimesEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- si la hora del primer punto no ha sido establecida, se colocará en la barra actu
   if(!time1)
      time1=TimeCurrent();
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
```

© 2000-2025, MetaQuotes Ltd.

```
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del segundo punto no ha sido establecida, se colocará a 2 barras a la
   if(!time2)
     {
      //--- array para recibir la hora de apertura de las últimas 3 barras
      datetime temp[3];
      CopyTime(Symbol(),Period(),time1,3,temp);
      //--- colocamos el primer punto a 2 barras a la izquierda del segundo
      time2=temp[0];
     }
//--- si el precio del segundo punto no ha sido establecido, va a coincidir con el pre
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje de las "Zon
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
```

© 2000-2025, MetaQuotes Ltd.

```
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar las "Zonas temporales de Fibonacci"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- creamos el objeto
   if(!FiboTimesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje
//--- contador del ciclo
   int h_steps=bars*2/5;
//--- movemos el segundo punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d2<bars-1)
         d2+=1;
      //--- movemos el punto
      if(!FiboTimesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   h_steps=bars*3/5;
//--- movemos el primer punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d1<bars-1)
         d1+=1;
      //--- movemos el punto
      if(!FiboTimesPointChange(0,InpName,0,date[d1],price[p1]))
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el objeto desde el gráfico
   FiboTimesDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOFAN

Abanico de Fibonacci.



Nota

Para el "Abanico de Fibonacci" se puede indicar el número de sus líneas-niveles, su valor y el color.

Ejemplo

El siguiente script crea y desplaza el "Abanico de Fibonacci" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Abanico de Fibonacci\".
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="FiboFan";          // Nombre del abanico
input int             InpDate1=10;                // Fecha del 1-er punto en %
input int             InpPrice1=25;               // Precio del 1-er punto en %
input int             InpDate2=30;                // Fecha del 2-do punto en %
input int             InpPrice2=50;               // Precio del 2-do punto en %
input color           InpColor=clrRed;            // Color de la líneas del abanico
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de la línea
input int             InpWidth=2;                 // Grosor de la línea
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpBack=false;             // Objeto al fondo
input bool            InpSelection=true;         // Seleccionar para mover
input bool            InpHidden=true;            // Ocultar en la lista de objetos
input long            InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el "Abanico de Fibonacci" según las coordenadas establecidas |
//+------------------------------------------------------------------+
bool FiboFanCreate(const long            chart_ID=0,        // ID del gráfico
                   const string          name="FiboFan",    // nombre del abanico
                   const int             sub_window=0,      // número de subventana
                   datetime              time1=0,           // hora del primer punto
                   double                price1=0,          // precio del primer punto
                   datetime              time2=0,           // hora del segundo punto
                   double                price2=0,          // precio del segundo punto
                   const color           clr=clrRed,        // color de la línea del a
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de la línea del
                   const int             width=1,           // grosor de la línea del
                   const bool            back=false,        // al fondo
                   const bool            selection=true,    // seleccionar para mover
                   const bool            hidden=true,       // ocultar en la lista de
                   const long            z_order=0)         // prioridad para el clic
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeFiboFanEmptyPoints(time1,price1,time2,price2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el "Abanico de Fibonacci" según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOFAN,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el \"Abanico de Fibonacci\"! Código del error = ",GetLa
      return(false);
     }
//--- establecemos el color
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del abanico para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Establece el número de los niveles y sus parámetros              |
//+------------------------------------------------------------------+
bool FiboFanLevelsSet(int             levels,         // número de las líneas del nive
                      double          &values[],      // valores de las líneas del niv
                      color           &colors[],      // color de las líneas del nivel
                      ENUM_LINE_STYLE &styles[],      // estilo de las líneas del nive
                      int             &widths[],      // grosor de las líneas del nive
                      const long      chart_ID=0,     // ID del gráfico
                      const string    name="FiboFan") // nombre del abanico
  {
//--- comprobamos los tamaños de los arrays
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": ¡Error. La longitud del array no corresponde al número de
      return(false);
     }
//--- establecemos el número de los niveles
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- establecemos las propiedades de los niveles en el ciclo
   for(int i=0;i<levels;i++)
     {
      //--- valor del nivel
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- color del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- grosor del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descripción del nivel
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje del "Abanico de Fibonacci"             |
//+------------------------------------------------------------------+
bool FiboFanPointChange(const long   chart_ID=0,     // ID del gráfico
                        const string name="FiboFan", // nombre del abanico
                        const int    point_index=0,  // número del punto de anclaje
                        datetime     time=0,         // coordenada del tiempo del punt
```

© 2000-2025, MetaQuotes Ltd.

```
                          double       price=0)        // coordenada del precio del punt
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el "Abanico de Fibonacci"                                |
//+------------------------------------------------------------------+
bool FiboFanDelete(const long   chart_ID=0,     // ID del gráfico
                   const string name="FiboFan") // nombre del abanico
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el Abanico
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el \"Abanico de Fibonacci\"! Código del error = ",Ge
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| //| Comprueba los valores de los puntos de anclaje de el "Abanico de Fibonacci" y
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeFiboFanEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- si la hora del segundo punto no ha sido establecida, se colocará en la barra act
   if(!time2)
      time2=TimeCurrent();
//--- si el precio del segundo punto no ha sido establecido, tendrá el valor Bid
   if(!price2)
```

© 2000-2025, MetaQuotes Ltd.

```
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del primer punto no ha sido establecida, se colocará a 9 barras a la
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
//--- si el precio del primer punto no ha sido establecido, lo moveremos a 200 puntos
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del "Abanic
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
```

© 2000-2025, MetaQuotes Ltd.

```
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el "Abanico de Fibonacci"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- creamos el objeto
   if(!FiboFanCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje del abanico
//--- contador del ciclo
   int v_steps=accuracy/2;
//--- movemos el primer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1<accuracy-1)
         p1+=1;
      //--- movemos el punto
      if(!FiboFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   int h_steps=bars/4;
//--- movemos el segundo punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d2<bars-1)
         d2+=1;
      //--- movemos el punto
      if(!FiboFanPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
```

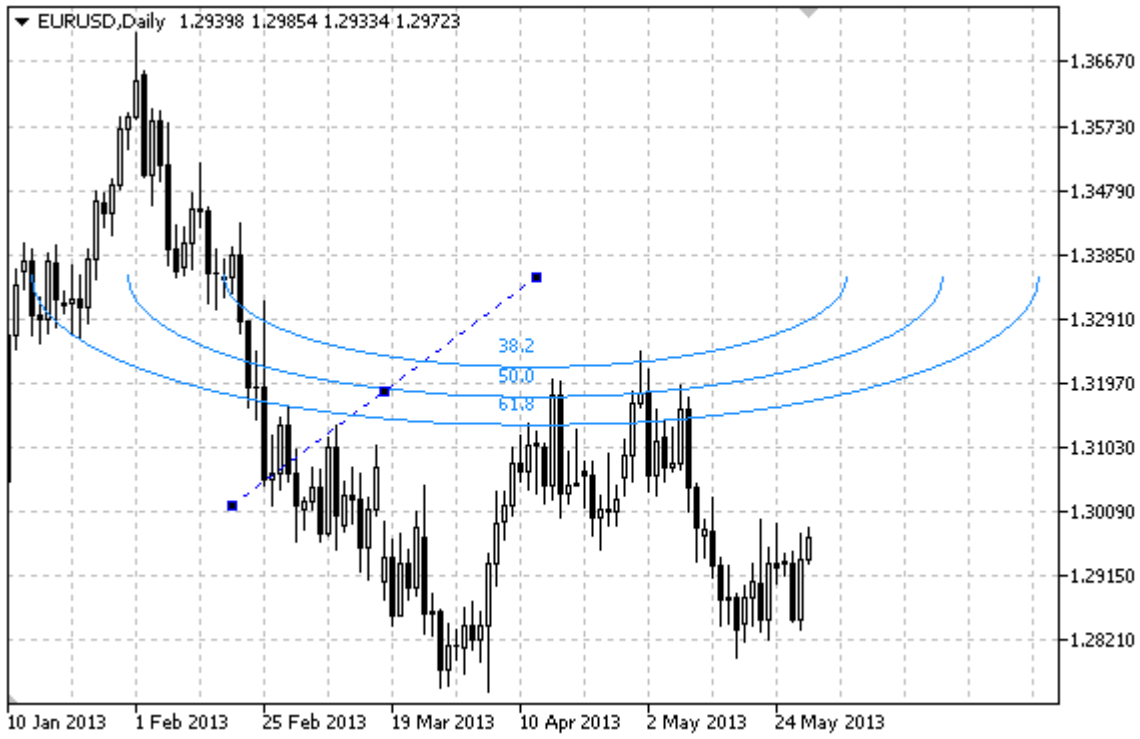
© 2000-2025, MetaQuotes Ltd.

```
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el objeto desde el gráfico
   FiboFanDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOARC

Arcos de Fibonacci.



Nota

Para los "Arcos de Fibonacci" se puede indicar el modo de visualización de la elipse entera. Cambiando la escala y las coordenadas de los puntos de anclaje se puede establecer el radio de la curvatura.

Además, se puede indicar el número de líneas-niveles, su valor y el color.

Ejemplo

El siguiente script crea y desplaza los "Arcos de Fibonacci" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Arcos de Fibonacci\"."
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="FiboArc";         // Nombre del objeto
input int             InpDate1=25;               // Fecha del 1-er punto en %
input int             InpPrice1=25;              // Precio del 1-er punto en %
input int             InpDate2=35;               // Fecha del 2-do punto en %
input int             InpPrice2=55;              // Precio del 2-do punto en %
```

© 2000-2025, MetaQuotes Ltd.

```
input double           InpScale=3.0;              // Escala
input bool             InpFullEllipse=true;       // Forma de los arcos
input color            InpColor=clrRed;           // Color de la línea
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Estilo de la línea
input int              InpWidth=2;                // Grosor de la línea
input bool             InpBack=false;             // Objeto al fondo
input bool             InpSelection=true;         // Seleccionar para mover
input bool             InpHidden=true;            // Ocultar en la lista de objetos
input long             InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea los "Arcos de Fibonacci" según las coordenadas establecidas |
//+------------------------------------------------------------------+
bool FiboArcCreate(const long            chart_ID=0,          // ID del gráfico
                   const string          name="FiboArc",      // nombre del objeto
                   const int             sub_window=0,        // número de subventana
                   datetime              time1=0,             // hora del primer punto
                   double                price1=0,            // precio del primer punto
                   datetime              time2=0,             // hora del segundo punto
                   double                price2=0,            // precio del segundo pur
                   const double          scale=1.0,           // escala
                   const bool            full_ellipse=false,  // forma de los arcos
                   const color           clr=clrRed,          // color de la línea
                   const ENUM_LINE_STYLE style=STYLE_SOLID,   // estilo de la línea
                   const int             width=1,             // grosor de la línea
                   const bool            back=false,          // al fondo
                   const bool            selection=true,      // seleccionar para mover
                   const bool            hidden=true,         // ocultar en la lista de
                   const long            z_order=0)           // prioridad para el clic
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeFiboArcEmptyPoints(time1,price1,time2,price2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos los "Arcos de Fibonacci" según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOARC,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": ¡Falo al crear los \"Arcos de Fibonacci\"! Código del error = ",GetLast
      return(false);
     }
//--- establecemos la escala
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- establecemos la visualización de los arcos en forma de la elipsis entera (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_ELLIPSE,full_ellipse);
//--- establecemos el color
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de la línea
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección de los arcos para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Establece el número de los niveles y sus parámetros              |
//+------------------------------------------------------------------+
bool FiboArcLevelsSet(int              levels,         // número de las líneas del nive
                      double           &values[],      // valores de las líneas del niv
                      color            &colors[],      // color de las líneas del nivel
                      ENUM_LINE_STYLE &styles[],       // estilo de las líneas del nive
                      int              &widths[],      // grosor de las líneas del nive
                      const long       chart_ID=0,     // ID del gráfico
                      const string     name="FiboArc") // nombre del objeto
  {
//--- comprobamos los tamaños de los arrays
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": ¡Error. La longitud del array no corresponde al número de
      return(false);
     }
//--- establecemos el número de los niveles
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- establecemos las propiedades de los niveles en el ciclo
   for(int i=0;i<levels;i++)
     {
      //--- valor del nivel
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- color del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- grosor del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descripción del nivel
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje de los "Arcos de Fibonacci"            |
//+------------------------------------------------------------------+
bool FiboArcPointChange(const long   chart_ID=0,     // ID del gráfico
                        const string name="FiboArc", // nombre del objeto
                        const int    point_index=0,  // número del punto de anclaje
                        datetime     time=0,         // coordenada del tiempo del punt
                        double       price=0)        // coordenada del precio del punt
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina los "Arcos de Fibonacci"                                 |
//+------------------------------------------------------------------+
bool FiboArcDelete(const long   chart_ID=0,     // ID del gráfico
                   const string name="FiboArc") // nombre de objetos
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Falo al eliminar los \"Arcos de Fibonacci\"! Código del error = ",GetL
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Comprueba los valores de los puntos de anclaje de los "Arcos de Fibonacci" y para
//| los valores vacíos establece los valores por defecto                     |
//+------------------------------------------------------------------+
void ChangeFiboArcEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- si la hora del segundo punto no ha sido establecida, se colocará en la barra act
   if(!time2)
      time2=TimeCurrent();
//--- si el precio del segundo punto no ha sido establecido, tendrá el valor Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del primer punto no ha sido establecida, se colocará a 9 barras a la
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
//--- si el precio del primer punto no ha sido establecido, lo moveremos a 300 puntos
   if(!price1)
      price1=price2-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje de los "Arc
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar e los "Arcos de Fibonacci"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- creamos el objeto
   if(!FiboArcCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpScale,
      InpFullEllipse,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrd
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje
//--- contador del ciclo
   int v_steps=accuracy/5;
//--- movemos el primer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1<accuracy-1)
         p1+=1;
      //--- movemos el punto
      if(!FiboArcPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
```

© 2000-2025, MetaQuotes Ltd.

```
   int h_steps=bars/5;
//--- movemos el segundo punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d2<bars-1)
         d2+=1;
      //--- movemos el punto
      if(!FiboArcPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el objeto desde el gráfico
   FiboArcDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOCHANNEL

Canal de Fibonacci.



Nota

Para el "Canal de Fibonacci" se puede indicar el modo de su continuación a la derecha y/o a la izquierda (propiedades OBJPROP_RAY_RIGHT y OBJPROP_RAY_LEFT respectivamente).

Además, se puede indicar el número de líneas-niveles, su valor y el color.

Ejemplo

El siguiente script crea y desplaza el "Canal de Fibonacci" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Canal de Fibonacci\"."
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="FiboChannel";     // Nombre del canal
input int             InpDate1=20;               // Fecha del 1-er punto en %
input int             InpPrice1=10;              // Precio del 1-er punto en %
input int             InpDate2=60;               // Fecha del 2-do punto en %
input int             InpPrice2=30;              // Precio del 2-do punto en %
input int             InpDate3=20;               // Fecha del 3-er punto en %
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpPrice3=25;              // Precio del 3-er punto en %
input color            InpColor=clrRed;           // Color del canal
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Estilo de las líneas del canal
input int              InpWidth=2;                // Grosor de las líneas del canal
input bool             InpBack=false;             // Canal al fondo
input bool             InpSelection=true;         // Seleccionar para mover
input bool             InpRayLeft=false;          // Continuación del canal a la izquie
input bool             InpRayRight=false;         // Continuación del canal a la derech
input bool             InpHidden=true;            // Ocultar en la lista de objetos
input long             InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el "Canal de Fibonacci" según las coordenadas establecidas  |
//+------------------------------------------------------------------+
bool FiboChannelCreate(const long             chart_ID=0,         // ID del gráfico
                       const string           name="FiboChannel", // nombre del canal
                       const int              sub_window=0,       // número de subventa
                     datetime               time1=0,            // hora del primer punto
                    double                 price1=0,           // precio del primer punto
                    datetime               time2=0,            // hora del segundo punto
                     double                 price2=0,           // precio del segundo punto
                       datetime               time3=0,            // hora del tercer pu
                       double                 price3=0,           // precio del tercer
                       const color            clr=clrRed,         // color del canal
                       const ENUM_LINE_STYLE  style=STYLE_SOLID,  // estilo de las líne
                       const int              width=1,            // grosor de las líne
                       const bool             back=false,         // al fondo
                       const bool             selection=true,     // seleccionar para m
                       const bool             ray_left=false,     // continuación del c
                       const bool             ray_right=false,    // continuación del c
                       const bool             hidden=true,        // ocultar en la list
                       const long             z_order=0)          // prioridad para el
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeFiboChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el canal según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOCHANNEL,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el \"Canal de Fibonacci\"! Código del error = ",GetLast
      return(false);
     }
//--- fijamos el color del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de las líneas del canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del canal para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activamos (true) o desactivamos (false) el modo de continuación del canal a la i
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activamos (true) o desactivamos (false) el modo de continuación del canal a la d
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Establece el número de los niveles y sus parámetros              |
//+------------------------------------------------------------------+
bool FiboChannelLevelsSet(int              levels,              // número de las líneas
                          double           &values[],           // valores de las líneas
                          color            &colors[],           // color de las líneas d
                          ENUM_LINE_STYLE  &styles[],           // estilo de las líneas
                          int              &widths[],           // grosor de las líneas
                          const long       chart_ID=0,          // ID del gráfico
                          const string     name="FiboChannel") // nombre del objeto
  {
//--- comprobamos los tamaños de los arrays
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": ¡Error. La longitud del array no corresponde al número de
      return(false);
     }
//--- establecemos el número de los niveles
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- establecemos las propiedades de los niveles en el ciclo
   for(int i=0;i<levels;i++)
     {
      //--- valor del nivel
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- color del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- grosor del nivel
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descripción del nivel
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje del "Canal de Fibonacci"               |
//+------------------------------------------------------------------+
bool FiboChannelPointChange(const long   chart_ID=0,         // ID del gráfico
                            const string name="FiboChannel", // nombre del canal
                            const int    point_index=0,      // número del punto de ar
                            datetime     time=0,             // coordenada del tiempo
                            double       price=0)            // coordenada del precio
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el canal                                                 |
//+------------------------------------------------------------------+
bool FiboChannelDelete(const long   chart_ID=0,         // ID del gráfico
                       const string name="FiboChannel") // nombre del canal
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el \"Canal de Fibonacci\"! Código del error = ",GetL
      return(false);
     }
//--- ejecución con éxito
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| //| Comprueba los valores de los puntos de anclaje de el "Canal de Fibonacci" y pa
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeFiboChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                  double &price2,datetime &time3,double &price3)
  {
//--- si la hora del segundo punto (de la derecha) no ha sido establecida, se colocará
   if(!time2)
      time2=TimeCurrent();
//--- si el precio del segundo punto no ha sido establecido, tendrá el valor Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del primer punto (de la izquierda) no ha sido establecida, se colocar
   if(!time1)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
//--- si el precio del primer punto no ha sido establecido, lo moveremos a 300 puntos
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si la hora del tercer punto no ha sido establecida, va a coincidir con la hora d
   if(!time3)
      time3=time1;
//--- si el precio del tercer punto no ha sido establecido, va a coincidir con el prec
   if(!price3)
      price3=price2;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del canal
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos puntos para trazar el canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- creamos el "Canal de Fibonacci"
   if(!FiboChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje del canal
//--- contador del ciclo
   int h_steps=bars/10;
//--- movemos el primer punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d1>1)
         d1-=1;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- movemos el punto
      if(!FiboChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   int v_steps=accuracy/10;
//--- movemos el segundo punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p2>1)
         p2-=1;
      //--- movemos el punto
      if(!FiboChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   v_steps=accuracy/15;
//--- movemos el tercer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p3<accuracy-1)
         p3+=1;
      //--- movemos el punto
      if(!FiboChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
```

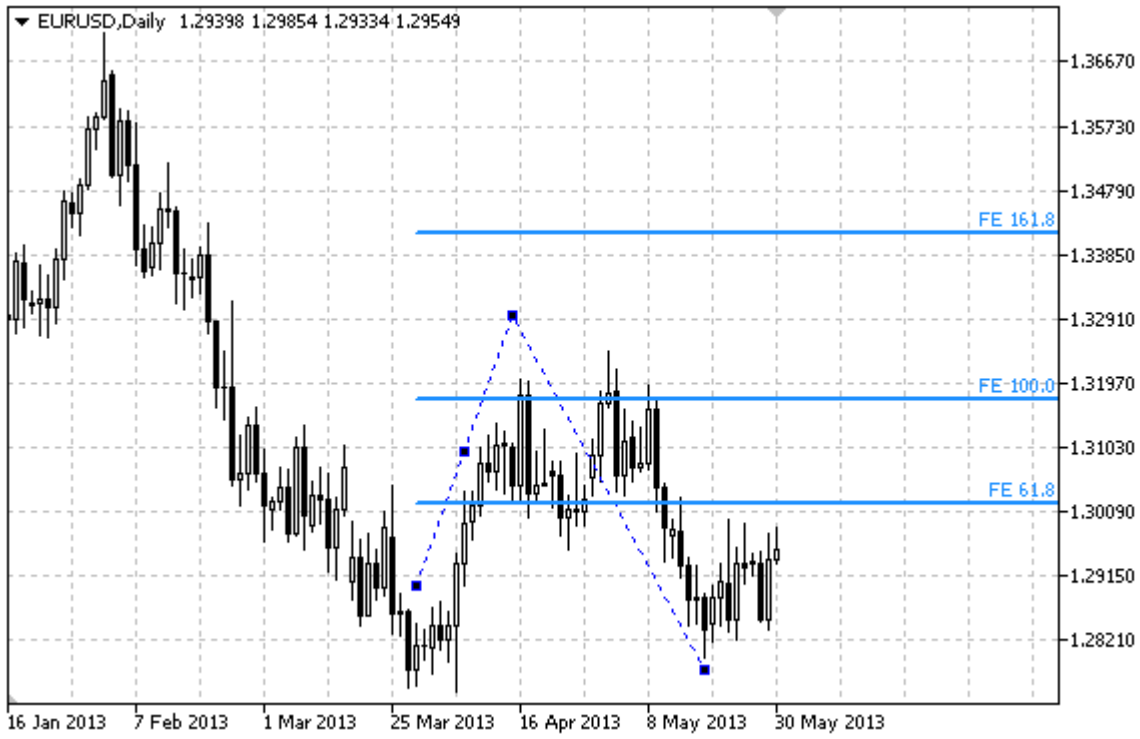
© 2000-2025, MetaQuotes Ltd.

```
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el canal desde el gráfico
   FiboChannelDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EXPANSION

Expansiones de Fibonacci.



Nota

Para las "Expansiones de Fibonacci" se puede indicar el modo de su continuación a la derecha y/o a la izquierda (propiedades OBJPROP_RAY_RIGHT y OBJPROP_RAY_LEFT respectivamente).

Además, se puede indicar el número de líneas-niveles, su valor y el color.

Ejemplo

El siguiente script crea y desplaza las "Expansiones de Fibonacci" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Expansiones de Fibonacc
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="FiboExpansion";   // Nombre del objeto
input int             InpDate1=10;               // Fecha del 1-er punto en %
input int             InpPrice1=55;              // Precio del 1-er punto en %
input int             InpDate2=30;               // Fecha del 2-do punto en %
input int             InpPrice2=10;              // Precio del 2-do punto en %
input int             InpDate3=80;               // Fecha del 3-er punto en %
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpPrice3=75;              // Precio del 3-er punto en %
input color            InpColor=clrRed;           // Color del objeto
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Estilo de la línea
input int              InpWidth=2;                // Grosor de la línea
input bool             InpBack=false;             // Objeto al fondo
input bool             InpSelection=true;         // Seleccionar para mover
input bool             InpRayLeft=false;          // Continuación del objeto a la izqui
input bool             InpRayRight=false;         // Continuación del objeto a la derec
input bool             InpHidden=true;            // Ocultar en la lista de objetos
input long             InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea las "Expansiones de Fibonacci" según las coordenadas establecidas
//+------------------------------------------------------------------+
bool FiboExpansionCreate(const long             chart_ID=0,           // ID del gráfico
                         const string           name="FiboExpansion", // nombre del car
                         const int              sub_window=0,         // número de subv
                         datetime               time1=0,              // hora del prime
                         double                 price1=0,             // precio del pri
                         datetime               time2=0,              // hora del segun
                         double                 price2=0,             // precio del seg
                         datetime               time3=0,              // hora del terce
                         double                 price3=0,             // precio del ter
                         const color            clr=clrRed,           // color del obje
                         const ENUM_LINE_STYLE  style=STYLE_SOLID,    // estilo de las
                         const int              width=1,              // grosor de las
                         const bool             back=false,           // al fondo
                         const bool             selection=true,       // seleccionar pa
                         const bool             ray_left=false,       // continuación d
                         const bool             ray_right=false,      // continuación d
                         const bool             hidden=true,          // ocultar en la
                         const long             z_order=0)            // prioridad para
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeFiboExpansionEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos las "Expansiones de Fibonacci" según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_EXPANSION,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": ¡Falo al crear las \"Expansiones de Fibonacci\"! Código del error = ",G
      return(false);
     }
//--- establecemos el color del objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- fijamos el grosor de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del objeto para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activamos (true) o desactivamos (false) el modo de continuación del objeto a la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activamos (true) o desactivamos (false) el modo de continuación del objeto a la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Establece el número de los niveles y sus parámetros              |
//+------------------------------------------------------------------+
bool FiboExpansionLevelsSet(int              levels,               // número de las lín
                            double           &values[],            // valores de las lí
                            color            &colors[],            // color de las líne
                            ENUM_LINE_STYLE  &styles[],            // estilos de las lí
                            int              &widths[],            // grosor de las lín
                            const long       chart_ID=0,           // ID del gráfico
                            const string     name="FiboExpansion") // nombre del objeto
  {
//--- comprobamos los tamaños de los arrays
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": ¡Error. La longitud del array no corresponde al número de
      return(false);
     }
//--- establecemos el número de los niveles
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- establecemos las propiedades de los niveles en el ciclo
   for(int i=0;i<levels;i++)
     {
      //--- valor del nivel
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- color del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo del nivel
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- grosor del nivel
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descripción del nivel
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,"FE "+DoubleToString(100*value
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje de las "Expansiones de Fibonacci"      |
//+------------------------------------------------------------------+
bool FiboExpansionPointChange(const long   chart_ID=0,           // ID del gráfico
                              const string name="FiboExpansion", // nombre del objeto
                              const int    point_index=0,        // número del punto d
                              datetime     time=0,               // coordenada del tie
                              double       price=0)              // coordenada del pre
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina las "Expansiones de Fibonacci"                           |
//+------------------------------------------------------------------+
bool FiboExpansionDelete(const long   chart_ID=0,           // ID del gráfico
                         const string name="FiboExpansion") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Falo al eliminar las \"Expansiones de Fibonacci\"! Código del error =
      return(false);
     }
//--- ejecución con éxito
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| //| Comprueba los valores de los puntos de anclaje de las "Expansiones de Fibonacc
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeFiboExpansionEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                    double &price2,datetime &time3,double &price3)
  {
//--- si la hora del tercer punto (de la derecha) no ha sido establecida, se colocará
   if(!time3)
      time3=TimeCurrent();
//--- si el precio del tercer punto no ha sido establecido, tendrá el valor Bid
   if(!price3)
      price3=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del primer punto (de la izquierda) no ha sido establecida, se colocar
//--- array para recibir la hora de apertura de las últimas 10 barras
   datetime temp[];
   ArrayResize(temp,10);
   if(!time1)
     {
      CopyTime(Symbol(),Period(),time3,10,temp);
      //--- colocamos el primer punto a 9 barras a la izquierda del segundo
      time1=temp[0];
     }
//--- si el precio del primer punto no ha sido establecido, va a coincidir con el prec
   if(!price1)
      price1=price3;
//--- si la hora del segundo punto no ha sido establecida, se colocará a 7 barras a la
   if(!time2)
      time2=temp[2];
//--- si el precio del segundo punto no ha sido establecido, lo moveremos a 250 puntos
   if(!price2)
      price2=price1-250*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del objeto
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar las "Expansiones de Fibonacci"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- creamos las "Expansiones de Fibonacci"
   if(!FiboExpansionCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje
//--- contador del ciclo
   int v_steps=accuracy/10;
//--- movemos el primer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1>1)
```

© 2000-2025, MetaQuotes Ltd.

```
         p1-=1;
       //--- movemos el punto
       if(!FiboExpansionPointChange(0,InpName,0,date[d1],price[p1]))
          return;
       //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
       if(IsStopped())
          return;
       //--- redibujamos el gráfico
       ChartRedraw();
      }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   v_steps=accuracy/2;
//--- movemos el tercer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p3>1)
         p3-=1;
      //--- movemos el punto
      if(!FiboExpansionPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   v_steps=accuracy*4/5;
//--- movemos el segundo punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p2<accuracy-1)
         p2+=1;
      //--- movemos el punto
      if(!FiboExpansionPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(1000);
//--- eliminamos el objeto desde el gráfico
   FiboExpansionDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIOTWAVE5

Onda de impulso de Elliott.



Nota

Para la "Onda de impulso de Elliott" se puede activar/desactivar el modo de unión de los puntos con líneas (propiedad OBJPROP_DRAWLINES), así como establecer el nivel de trazado ondular (desde la enumeración ENUM_ELLIOT_WAVE_DEGREE).

Ejemplo

El siguiente script crea y desplaza la "Onda de impulso de Elliott" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Onda de impulso de Elli
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string                InpName="ElliotWave5";   // Nombre del objeto
input int                   InpDate1=10;             // Fecha del 1-er punto en %
input int                   InpPrice1=90;            // Precio del 1-er punto en %
input int                   InpDate2=20;             // Fecha del 2-do punto en %
input int                   InpPrice2=40;            // Precio del 2-do punto en %
input int                   InpDate3=30;             // Fecha del 3-er punto en %
```

© 2000-2025, MetaQuotes Ltd.

```
input int                      InpPrice3=60;            // Precio del 3-er punto en %
input int                      InpDate4=40;             // Fecha del 4-to punto en %
input int                      InpPrice4=10;            // Precio del 4-to punto en %
input int                      InpDate5=60;             // Fecha del 5-to punto en %
input int                      InpPrice5=40;            // Precio del 5-to punto en %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Nivel
input bool                     InpDrawLines=true;       // Mostrar líneas
input color                    InpColor=clrRed;         // Color de las líneas
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // Estilo de las líneas
input int                      InpWidth=2;              // Grosor de las líneas
input bool                     InpBack=false;           // Objeto al fondo
input bool                     InpSelection=true;       // Seleccionar para mover
input bool                     InpHidden=true;          // Ocultar en la lista de objet
input long                     InpZOrder=0;             // Prioridad para el clic del r
//+------------------------------------------------------------------+
//| Crea la "Onda de impulso de Elliott" según las coordinadas establecidas       |
//+------------------------------------------------------------------+
bool ElliotWave5Create(const long                  chart_ID=0,              // ID de
                       const string                name="ElliotWave5",      // nombr
                       const int                   sub_window=0,            // núme
                       datetime                    time1=0,                 // hora
                       double                      price1=0,                // preci
                       datetime                    time2=0,                 // hora
                       double                      price2=0,                // preci
                       datetime                    time3=0,                 // hora
                       double                      price3=0,                // preci
                       datetime                    time4=0,                 // hora
                       double                      price4=0,                // preci
                       datetime                    time5=0,                 // hora
                       double                      price5=0,                // preci
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // grado
                       const bool                  draw_lines=true,         // mostr
                       const color                 clr=clrRed,              // color
                       const ENUM_LINE_STYLE       style=STYLE_SOLID,       // estil
                       const int                   width=1,                 // gros
                       const bool                  back=false,              // al fo
                       const bool                  selection=true,          // selec
                       const bool                  hidden=true,             // ocult
                       const long                  z_order=0)               // prior
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeElliotWave5EmptyPoints(time1,price1,time2,price2,time3,price3,time4,price4,ti
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la "Onda de impulso de Elliott" según las coordinadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE5,sub_window,time1,price1,time2,price2
      price3,time4,price4,time5,price5))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": ¡Falo al crear la \"Onda de impulso de Elliott\"! Código del error = ",
      return(false);
     }
//--- establecemos el grado (tamaño de la onda)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- activar (true) o desactivar (false) el modo de visualización de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- establecemos el color del objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- fijamos el grosor de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del objeto para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje de la "Onda de impulso de Elliott"     |
//+------------------------------------------------------------------+
bool ElliotWave5PointChange(const long   chart_ID=0,          // ID del gráfico
                            const string name="ElliotWave5", // nombre del objeto
                            const int    point_index=0,      // número del punto de ar
                            datetime     time=0,             // coordenada del tiempo
                            double       price=0)            // coordenada del precio
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina la "Onda de impulso de Elliott"                          |
//+------------------------------------------------------------------+
bool ElliotWave5Delete(const long   chart_ID=0,         // ID del gráfico
                       const string name="ElliotWave5") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Falo al eliminar la \"Onda de impulso de Elliott\"! Código del error =
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| //| Comprueba los valores de los puntos de anclaje de la "Onda de impulso de Ellio
//| para los valores vacíos establece los valores por defecto        |
//+------------------------------------------------------------------+
void ChangeElliotWave5EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3,
                                  datetime &time4,double &price4,
                                  datetime &time5,double &price5)
  {
//--- array para recibir la hora de apertura de las últimas 10 barras
   datetime temp[];
   ArrayResize(temp,10);
//--- recibimos los datos
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- obtenemos el valor de un punto en al gráfico actual
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si la hora del primer punto no ha sido establecida, se colocará en la 9 barra a
   if(!time1)
      time1=temp[0];
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del segundo punto no ha sido establecida, se colocará en la 7 barra a
   if(!time2)
      time2=temp[2];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si el precio del segundo punto no ha sido establecido, lo moveremos a 300 puntos
   if(!price2)
      price2=price1-300*point;
//--- si la hora del tercer punto no ha sido establecida, se colocará en la 5 barra a
   if(!time3)
      time3=temp[4];
//--- si el precio del tercer punto no ha sido establecido, lo moveremos a 250 puntos
   if(!price3)
      price3=price1-250*point;
//--- si la hora del cuarto punto no ha sido establecida, se colocará en la 3 barra a
   if(!time4)
      time4=temp[6];
//--- si el precio del cuarto punto no ha sido establecido, lo moveremos a 550 puntos
   if(!price4)
      price4=price1-550*point;
//--- si la hora del quinto punto no ha sido establecida, se colocará en la última bar
   if(!time5)
      time5=temp[9];
//--- si el precio del quinto punto no ha sido establecido, lo moveremos a 450 puntos
   if(!price5)
      price5=price1-450*point;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100 ||
      InpDate4<0 || InpDate4>100 || InpPrice4<0 || InpPrice4>100 ||
      InpDate5<0 || InpDate5>100 || InpPrice5<0 || InpPrice5>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del objeto
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar la "Onda de impulso de Elliott"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int d4=InpDate4*(bars-1)/100;
   int d5=InpDate5*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
   int p4=InpPrice4*(accuracy-1)/100;
   int p5=InpPrice5*(accuracy-1)/100;
//--- creamos la "Onda de impulso de Elliott"
   if(!ElliotWave5Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      date[d4],price[p4],date[d5],price[p5],InpDegree,InpDrawLines,InpColor,InpStyle,I
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje
//--- contador del ciclo
   int v_steps=accuracy/5;
//--- movemos el quinto punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p5<accuracy-1)
         p5+=1;
      //--- movemos el punto
      if(!ElliotWave5PointChange(0,InpName,4,date[d5],price[p5]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
       //--- redibujamos el gráfico
       ChartRedraw();
      }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   v_steps=accuracy/5;
//--- movemos el segundo y el tercer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
       //--- cogemos los siguientes valores
       if(p2<accuracy-1)
          p2+=1;
       if(p3>1)
          p3-=1;
       //--- movemos los puntos
       if(!ElliotWave5PointChange(0,InpName,1,date[d2],price[p2]))
          return;
       if(!ElliotWave5PointChange(0,InpName,2,date[d3],price[p3]))
          return;
       //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
       if(IsStopped())
          return;
       //--- redibujamos el gráfico
       ChartRedraw();
      }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   v_steps=accuracy*4/5;
//--- movemos el primero y el cuarto punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
       //--- cogemos los siguientes valores
       if(p1>1)
          p1-=1;
       if(p4<accuracy-1)
          p4+=1;
       //--- movemos los puntos
       if(!ElliotWave5PointChange(0,InpName,0,date[d1],price[p1]))
          return;
       if(!ElliotWave5PointChange(0,InpName,3,date[d4],price[p4]))
          return;
       //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
       if(IsStopped())
          return;
       //--- redibujamos el gráfico
       ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el objeto desde el gráfico
   ElliotWave5Delete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIOTWAVE3

Onda correctiva de Elliott.



Nota

Para la "Onda correctiva de Elliott" se puede activar/desactivar el modo de unión de los puntos con líneas (propiedad OBJPROP_DRAWLINES), así como establecer el nivel de trazado ondular (desde la enumeración ENUM_ELLIOT_WAVE_DEGREE).

Ejemplo

El siguiente script crea y desplaza la "Onda de correctiva de Elliott" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Onda correctiva de Elli
#property description "Las coordenadas de los puntos de anclaje se establecen en por c
#property description "del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string                InpName="ElliotWave3";   // Nombre del objeto
input int                   InpDate1=10;             // Fecha del 1-er punto en %
input int                   InpPrice1=90;            // Precio del 1-er punto en %
input int                   InpDate2=30;             // Fecha del 2-do punto en %
input int                   InpPrice2=10;            // Precio del 2-do punto en %
input int                   InpDate3=50;             // Fecha del 3-er punto en %
```

© 2000-2025, MetaQuotes Ltd.

```
input int                      InpPrice3=40;            // Precio del 3-er punto en %
input ENUM_ELLIOT_WAVE_DEGREE  InpDegree=ELLIOTT_MINOR; // Nivel
input bool                     InpDrawLines=true;       // Mostrar líneas
input color                    InpColor=clrRed;         // Color de las líneas
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // Estilo de las líneas
input int                      InpWidth=2;              // Grosor de las líneas
input bool                     InpBack=false;           // Objeto al fondo
input bool                     InpSelection=true;       // Seleccionar para mover
input bool                     InpHidden=true;          // Ocultar en la lista de objet
input long                     InpZOrder=0;             // Prioridad para el clic del r
//+------------------------------------------------------------------+
//| Crea la "Onda correctiva de Elliott" según las coordinadas establecidas       |
//+------------------------------------------------------------------+
bool ElliotWave3Create(const long                   chart_ID=0,              // ID de
                       const string                 name="ElliotWave3",      // nombr
                       const int                    sub_window=0,            // núme
                       datetime                     time1=0,                 // hora
                       double                       price1=0,                // preci
                       datetime                     time2=0,                 // hora
                       double                       price2=0,                // preci
                       datetime                     time3=0,                 // hora
                       double                       price3=0,                // preci
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // grado
                       const bool                   draw_lines=true,         // mostr
                       const color                  clr=clrRed,              // color
                       const ENUM_LINE_STYLE        style=STYLE_SOLID,       // estil
                       const int                    width=1,                 // grosc
                       const bool                   back=false,              // al fo
                       const bool                   selection=true,          // selec
                       const bool                   hidden=true,             // ocult
                       const long                   z_order=0)               // prior
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeElliotWave3EmptyPoints(time1,price1,time2,price2,time3,price3);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la "Onda correctiva de Elliott" según las coordinadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE3,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": ¡Falo al crear la \"Onda correctiva de Elliott\"! Código del error = ",
      return(false);
     }
//--- establecemos el grado (tamaño de la onda)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- activar (true) o desactivar (false) el modo de visualización de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- establecemos el color del objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- establecemos el estilo de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- fijamos el grosor de las líneas
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del objeto para mover
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje de la "Onda correctiva de Elliott"     |
//+------------------------------------------------------------------+
bool ElliotWave3PointChange(const long   chart_ID=0,          // ID del gráfico
                            const string name="ElliotWave3", // nombre del objeto
                            const int    point_index=0,      // número del punto de ar
                            datetime     time=0,             // coordenada del tiempo
                            double       price=0)            // coordenada del precio
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina la "Onda correctiva de Elliott"                          |
//+------------------------------------------------------------------+
bool ElliotWave3Delete(const long   chart_ID=0,          // ID del gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
                        const string name="ElliotWave3") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Falo al eliminar la \"Onda correctiva de Elliott\"! Código del error =
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| //| Comprueba los valores de los puntos de anclaje de la "Onda correctiva de Ellio
//| para los valores vacíos establece los valores por defecto                |
//+------------------------------------------------------------------+
void ChangeElliotWave3EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3)
  {
//--- array para recibir la hora de apertura de las últimas 10 barras
   datetime temp[];
   ArrayResize(temp,10);
//--- recibimos los datos
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- obtenemos el valor de un punto en al gráfico actual
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si la hora del primer punto no ha sido establecida, se colocará en la 9 barra a
   if(!time1)
      time1=temp[0];
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del segundo punto no ha sido establecida, se colocará en la 5 barra a
   if(!time2)
      time2=temp[4];
//--- si el precio del segundo punto no ha sido establecido, lo moveremos a 300 puntos
   if(!price2)
      price2=price1-300*point;
//--- si la hora del tercer punto no ha sido establecida, se colocará en la 1 barra a
   if(!time3)
      time3=temp[8];
//--- si el precio del tercer punto no ha sido establecido, lo moveremos a 200 puntos
   if(!price3)
      price3=price1-200*point;
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del objeto
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar la "Onda correctiva de Elliott"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- creamos la "Onda correctiva de Elliott"
   if(!ElliotWave3Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      InpDegree,InpDrawLines,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidder
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje
//--- contador del ciclo
   int v_steps=accuracy/5;
//--- movemos el tercer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p3<accuracy-1)
         p3+=1;
      //--- movemos el punto
      if(!ElliotWave3PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   v_steps=accuracy*4/5;
//--- movemos el primero y el segundo punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos los siguientes valores
      if(p1>1)
         p1-=1;
      if(p2<accuracy-1)
         p2+=1;
      //--- movemos los puntos
      if(!ElliotWave3PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave3PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el objeto desde el gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
   ElliotWave3Delete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_RECTANGLE

Rectángulo.



Nota

Para el rectángulo se puede establecer el modo de relleno usando la propiedad OBJPROP_FILL.

Ejemplo

El siguiente script crea y desplaza el rectángulo en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el rectángulo en el gráfico."
#property description "Las coordenadas de los puntos de anclaje se establecen en"
#property description "por cientos de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="Rectangle"; // Nombre del rectángulo
input int             InpDate1=40;         // Fecha del 1-er punto en %
input int             InpPrice1=40;        // Precio del 1-er punto en %
input int             InpDate2=60;         // Fecha del 2-do punto en %
input int             InpPrice2=60;        // Precio del 2-do punto en %
input color           InpColor=clrRed;     // Color del rectángulo
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Estilo de las líneas del rectángulo
input int             InpWidth=2;          // Grosor de las líneas del rectángulo
```

© 2000-2025, MetaQuotes Ltd.

```
input bool             InpFill=true;        // Relleno del rectángulo con el color
input bool             InpBack=false;       // Rectángulo al fondo
input bool             InpSelection=true;   // Seleccionar para mover
input bool             InpHidden=true;      // Ocultar en la lista de objetos
input long             InpZOrder=0;         // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el rectángulo según las coordenadas establecidas            |
//+------------------------------------------------------------------+
bool RectangleCreate(const long            chart_ID=0,        // ID del gráfico
                     const string          name="Rectangle",  // nombre del rectángulo
                     const int             sub_window=0,      // número de subventana
                     datetime              time1=0,           // hora del primer punto
                     double                price1=0,          // precio del primer pun
                     datetime              time2=0,           // hora del segundo punt
                     double                price2=0,          // precio del segundo pu
                     const color           clr=clrRed,        // color del rectángulo
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de las líneas
                     const int             width=1,           // grosor de las líneas
                     const bool            fill=false,        // relleno del rectángul
                     const bool            back=false,        // al fondo
                     const bool            selection=true,    // seleccionar para move
                     const bool            hidden=true,       // ocultar en la lista d
                     const long            z_order=0)         // prioridad para el cli
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeRectangleEmptyPoints(time1,price1,time2,price2);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el rectángulo según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el rectángulo! Código del error = ",GetLastError());
      return(false);
     }
//--- establecemos el color del rectángulo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas del rectángulo
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de las líneas del rectángulo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activar (true) o desactivar (false) el modo de relleno del rectángulo
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección del rectángulo para mov
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje del rectángulo                         |
//+------------------------------------------------------------------+
bool RectanglePointChange(const long   chart_ID=0,       // ID del gráfico
                          const string name="Rectangle", // nombre del rectángulo
                          const int    point_index=0,    // número del punto de anclaj
                          datetime     time=0,           // coordenada del tiempo del
                          double       price=0)          // coordenada del precio del
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el rectángulo                                            |
//+------------------------------------------------------------------+
bool RectangleDelete(const long   chart_ID=0,       // ID del gráfico
                     const string name="Rectangle") // nombre del rectángulo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el rectángulo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el rectángulo! Código del error = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje del rectángulo y para      |
//| los valores vacíos establece los valores por defecto                      |
//+------------------------------------------------------------------+
void ChangeRectangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- si la hora del primer punto no ha sido establecida, se colocará en la barra actu
   if(!time1)
      time1=TimeCurrent();
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del segundo punto no ha sido establecida, se colocará a 9 barras a la
   if(!time2)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- colocamos el segundo punto a 9 barras a la izquierda del primero
      time2=temp[0];
     }
//--- si el precio del segundo punto no ha sido establecido, lo moveremos a 300 puntos
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del rectáng
   datetime date[];
```

© 2000-2025, MetaQuotes Ltd.

```
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el rectángulo
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- creamos el rectángulo
   if(!RectangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje del rectángulo
//--- contador del ciclo
   int h_steps=bars/2;
//--- movemos los puntos de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos los siguientes valores
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- movemos los puntos
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   int v_steps=accuracy/2;
//--- movemos los puntos de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos los siguientes valores
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- movemos los puntos
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el rectángulo desde el gráfico
   RectangleDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TRIANGLE

Triángulo.



Nota

Para el triángulo se puede establecer el modo de relleno usando la propiedad OBJPROP_FILL.

Ejemplo

El siguiente script crea y desplaza el triángulo en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el triángulo en el gráfico."
#property description "Las coordenadas de los puntos de anclaje se establecen en"
#property description "por cientos de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="Triangle";        // Nombre del triángulo
input int             InpDate1=25;               // Fecha del 1-er punto en %
input int             InpPrice1=50;              // Precio del 1-er punto en %
input int             InpDate2=70;               // Fecha del 2-do punto en %
input int             InpPrice2=70;              // Precio del 2-do punto en %
input int             InpDate3=65;               // Fecha del 3-er punto en %
input int             InpPrice3=20;              // Precio del 3-er punto en %
input color           InpColor=clrRed;           // Color del triángulo
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de las líneas del triángulo
input int             InpWidth=2;                // Grosor de las líneas del triángulo
input bool            InpFill=false;             // Relleno del triángulo con el color
input bool            InpBack=false;             // Triángulo al fondo
input bool            InpSelection=true;         // Seleccionar para mover
input bool            InpHidden=true;            // Ocultar en la lista de objetos
input long            InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el triángulo según las coordenadas establecidas             |
//+------------------------------------------------------------------+
bool TriangleCreate(const long            chart_ID=0,        // ID del gráfico
                    const string          name="Triangle",   // nombre del triángulo
                    const int             sub_window=0,      // número de subventana
                    datetime              time1=0,           // hora del primer punto
                    double                price1=0,          // precio del primer punt
                    datetime              time2=0,           // hora del segundo punto
                    double                price2=0,          // precio del segundo pun
                    datetime              time3=0,           // hora del segundo punto
                    double                price3=0,          // precio del tercer punt
                    const color           clr=clrRed,        // color del triángulo
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de las líneas d
                    const int             width=1,           // grosor de las líneas d
                    const bool            fill=false,        // relleno del triángulo
                    const bool            back=false,        // al fondo
                    const bool            selection=true,    // seleccionar para mover
                    const bool            hidden=true,       // ocultar en la lista de
                    const long            z_order=0)         // prioridad para el clic
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeTriangleEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el triángulo según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_TRIANGLE,sub_window,time1,price1,time2,price2,ti
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el triángulo! Código del error = ",GetLastError());
      return(false);
     }
//--- establecemos el color del triángulo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas del triángulo
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de las líneas del triángulo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activar (true) o desactivar (false) el modo de relleno del triángulo
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- activar (true) o desactivar (false) el modo de selección del triángulo para move
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje del triángulo                          |
//+------------------------------------------------------------------+
bool TrianglePointChange(const long   chart_ID=0,      // ID del gráfico
                         const string name="Triangle", // nombre del triángulo
                         const int    point_index=0,   // número del punto de anclaje
                         datetime     time=0,          // coordenada del tiempo del pu
                         double       price=0)         // coordenada del precio del pu
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el triángulo                                             |
//+------------------------------------------------------------------+
bool TriangleDelete(const long   chart_ID=0,      // ID del gráfico
                    const string name="Triangle") // nombre del triángulo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el triángulo
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": ¡Fallo al eliminar el triángulo! Código del error = ",GetLastError());
       return(false);
      }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje del triángulo y para      |
//| los valores vacíos establece los valores por defecto                     |
//+------------------------------------------------------------------+
void ChangeTriangleEmptyPoints(datetime &time1,double &price1,
                               datetime &time2,double &price2,
                               datetime &time3,double &price3)
  {
//--- si la hora del primer punto no ha sido establecida, se colocará en la barra actu
   if(!time1)
      time1=TimeCurrent();
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del segundo punto no ha sido establecida, se colocará a 9 barras a la
   if(!time2)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- colocamos el segundo punto a 9 barras a la izquierda del primero
      time2=temp[0];
     }
//--- si el precio del segundo punto no ha sido establecido, lo moveremos a 300 puntos
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si la hora del tercer punto no ha sido establecida, va a coincidir con la fecha
   if(!time3)
      time3=time2;
//--- si el precio del primer punto no ha sido establecido, va a coincidir con el prec
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
```

© 2000-2025, MetaQuotes Ltd.

```
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje del triángu
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el triángulo
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- creamos el triángulo
   if(!TriangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje del triángulo
//--- contador del ciclo
```

© 2000-2025, MetaQuotes Ltd.

```
   int v_steps=accuracy*3/10;
//--- movemos el primer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p1>1)
         p1-=1;
      //--- movemos el punto
      if(!TrianglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   int h_steps=bars*9/20-1;
//--- movemos el segundo punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d2>1)
         d2-=1;
      //--- movemos el punto
      if(!TrianglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   v_steps=accuracy/4;
//--- movemos el tercer punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p3<accuracy-1)
         p3+=1;
      //--- movemos el punto
      if(!TrianglePointChange(0,InpName,2,date[d3],price[p3]))
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el triángulo desde el gráfico
   TriangleDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIPSE

Elipse.



Nota

Para la elipse se puede establecer el modo de relleno usando la propiedad OBJPROP_FILL.

Ejemplo

El siguiente script crea y desplaza la elipse en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye la elipse en el gráfico."
#property description "Las coordenadas de los puntos de anclaje se establecen"
#property description "en por cientos de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="Ellipse";         // Nombre de la elipse
input int             InpDate1=30;               // Fecha del 1-er punto en %
input int             InpPrice1=20;              // Precio del 1-er punto en %
input int             InpDate2=70;               // Fecha del 2-do punto en %
input int             InpPrice2=80;              // Precio del 2-do punto en %
input int             InpDate3=50;               // Fecha del 3-er punto en %
input int             InpPrice3=60;              // Precio del 3-er punto en %
input color           InpColor=clrRed;           // Color de la elipse
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de las líneas de la elipse
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpWidth=2;                // Grosor de las líneas de la elipse
input bool             InpFill=false;             // Relleno de la elipse con el color
input bool             InpBack=false;             // La elipse al fondo
input bool             InpSelection=true;         // Seleccionar para mover
input bool             InpHidden=true;            // Ocultar en la lista de objetos
input long             InpZOrder=0;               // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea la elipse según las coordenadas establecidas                |
//+------------------------------------------------------------------+
bool EllipseCreate(const long            chart_ID=0,        // ID del gráfico
                   const string          name="Ellipse",    // nombre de la elipse
                   const int             sub_window=0,      // número de subventana
                   datetime              time1=0,           // hora del primer punto
                   double                price1=0,          // precio del primer punto
                   datetime              time2=0,           // hora del segundo punto
                   double                price2=0,          // precio del segundo punto
                   datetime              time3=0,           // hora del tercer punto
                   double                price3=0,          // precio del tercer punto
                   const color           clr=clrRed,        // color de la elipse
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de las líneas de
                   const int             width=1,           // grosor de las líneas de
                   const bool            fill=false,        // relleno de la elipse co
                   const bool            back=false,        // al fondo
                   const bool            selection=true,    // seleccionar para mover
                   const bool            hidden=true,       // ocultar en la lista de
                   const long            z_order=0)         // prioridad para el clic
  {
//--- establecemos las coordenadas de los puntos de anclaje si todavía no han sido est
   ChangeEllipseEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la elipse según las coordenadas establecidas
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIPSE,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la elipse! Código del error = ",GetLastError());
      return(false);
     }
//--- establecemos el color de la elipse
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas de la elipse
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor de las líneas de la elipse
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activar (true) o desactivar (false) el modo de relleno de la elipse
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de selección de la elipse para mover
```

© 2000-2025, MetaQuotes Ltd.

```
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje de la elipse                           |
//+------------------------------------------------------------------+
bool EllipsePointChange(const long   chart_ID=0,     // ID del gráfico
                        const string name="Ellipse", // nombre de la elipse
                        const int    point_index=0,  // número del punto de anclaje
                        datetime     time=0,         // coordenada del tiempo del punt
                        double       price=0)        // coordenada del precio del punt
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina la elipse                                                |
//+------------------------------------------------------------------+
bool EllipseDelete(const long   chart_ID=0,     // ID del gráfico
                   const string name="Ellipse") // nombre de la elipse
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la elipse
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la elipse! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje de la elipse y para   |
//| los valores vacíos establece los valores por defecto
//+------------------------------------------------------------------+
void ChangeEllipseEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2,
                              datetime &time3,double &price3)
  {
//--- si la hora del primer punto no ha sido establecida, se colocará en la barra actu
   if(!time1)
      time1=TimeCurrent();
//--- si el precio del primer punto no ha sido establecido, tendrá el valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si la hora del segundo punto no ha sido establecida, se colocará a 9 barras a la
   if(!time2)
     {
      //--- array para recibir la hora de apertura de las últimas 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- colocamos el segundo punto a 9 barras a la izquierda del primero
      time2=temp[0];
     }
//--- si el precio del segundo punto no ha sido establecido, lo moveremos a 300 puntos
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si la hora del tercer punto no ha sido establecida, va a coincidir con la fecha
   if(!time3)
      time3=time2;
//--- si el precio del tercer punto no ha sido establecido, va a coincidir con el prec
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas de los puntos de anclaje de la elips
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar la elipse
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- creamos la elipse
   if(!EllipseCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover los puntos de anclaje de la elipse
//--- contador del ciclo
   int v_steps=accuracy/5;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- movemos el primero y el segundo punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos los siguientes valores
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- movemos los puntos
      if(!EllipsePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!EllipsePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   int h_steps=bars/5;
//--- movemos el tercer punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d3>1)
         d3-=1;
      //--- movemos el punto
      if(!EllipsePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos la elipse desde el gráfico
   EllipseDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_THUMB_UP

Signo "Bien".



Nota

Desde la enumeración ENUM_ARROW_ANCHOR se puede elegir la posición del punto de anclaje respecto al signo.

Los signos de gran tamaño (más de 5) se puede crear sólo poniendo el correspondiente valor de la propiedad OBJPROP_WIDTH durante la escritura del código en MetaEditor.

Ejemplo

El siguiente script crea y desplaza el signo "Bien" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script dibuja el signo \"Bien\" (pulgar arriba)."
#property description "La coordenada del punto de anclaje se establece en por cientos"
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="ThumbUp";     // nombre del signo
input int               InpDate=75;            // Fecha del punto de anclaje en %
input int               InpPrice=25;           // Precio del punto de anclaje en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // Modo de anclaje
input color             InpColor=clrRed;       // Color del signo
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Estilo de la línea del contorno
input int               InpWidth=5;           // Tamaño del signo
input bool              InpBack=false;        // Signo al fondo
input bool              InpSelection=true;    // Seleccionar para mover
input bool              InpHidden=true;       // Ocultar en la lista de objetos
input long              InpZOrder=0;          // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el signo "Bien"                                             |
//+------------------------------------------------------------------+
bool ArrowThumbUpCreate(const long              chart_ID=0,           // ID del gráfic
                        const string            name="ThumbUp",       // nombre del si
                        const int               sub_window=0,         // número de sub
                        datetime                time=0,               // hora del punt
                        double                  price=0,              // precio del pu
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // modo de ancla
                        const color             clr=clrRed,           // color del sig
                        const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo de la
                        const int               width=3,              // tamaño del si
                        const bool              back=false,           // al fondo
                        const bool              selection=true,       // seleccionar p
                        const bool              hidden=true,          // ocultar en la
                        const long              z_order=0)            // prioridad par
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el signo
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el signo \"Bien\"! Código del error = ",GetLastError())
      return(false);
     }
//--- establecemos el modo de anclaje
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- establecemos el color del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea del contorno
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento del signo con rató
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool ArrowThumbUpMove(const long   chart_ID=0,      // ID del gráfico
                      const string name="ThumbUp", // nombre del objeto
                      datetime     time=0,         // coordenada del tiempo del punto
                      double       price=0)        // coordenada del precio del punto
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el modo de anclaje del signo "Bien"                       |
//+------------------------------------------------------------------+
bool ArrowThumbUpAnchorChange(const long              chart_ID=0,        // ID del grá
                              const string            name="ThumbUp",    // nombre del
                              const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // modo de an
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el modo de anclaje
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el modo de anclaje! Código del error = ",GetLastErro
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el signo "Bien"                                          |
//+------------------------------------------------------------------+
bool ArrowThumbUpDelete(const long   chart_ID=0,     // ID del gráfico
                        const string name="ThumbUp") // nombre del signo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el signo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el signo \"Bien\"! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas del punto de anclaje del signo
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el signo
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- creamos el signo "Bien" en el gráfico
   if(!ArrowThumbUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el punto de anclaje y cambiar su posición respecto al signo
//--- contador del ciclo
   int h_steps=bars/4;
//--- movemos el punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d>1)
         d-=1;
      //--- movemos el punto
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   int v_steps=accuracy/4;
//--- movemos el punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p<accuracy-1)
         p+=1;
      //--- movemos el punto
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- cambiamos la posición del punto de anclaje respecto al signo
   ArrowThumbUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redibujamos el gráfico
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el signo desde el gráfico
   ArrowThumbUpDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_THUMB_DOWN

Signo "Mal".



Nota

Desde la enumeración ENUM_ARROW_ANCHOR se puede elegir la posición del punto de anclaje respecto al signo.

Los signos de gran tamaño (más de 5) se puede crear sólo poniendo el correspondiente valor de la propiedad OBJPROP_WIDTH durante la escritura del código en MetaEditor.

Ejemplo

El siguiente script crea y desplaza el signo "Mal" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script dibuja el signo \"Mal\" (pulgar abajo)."
#property description "La coordenada del punto de anclaje se establece en por cientos'
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="ThumbDown";      // Nombre del signo
input int               InpDate=25;               // Fecha del punto de anclaje en %
input int               InpPrice=75;              // Precio del punto de anclaje en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Modo de anclaje
```

© 2000-2025, MetaQuotes Ltd.

```
input color             InpColor=clrRed;         // Color del signo
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Estilo de la línea del contorno
input int               InpWidth=5;              // Tamaño del signo
input bool              InpBack=false;           // Signo al fondo
input bool              InpSelection=true;       // Seleccionar para mover
input bool              InpHidden=true;          // Ocultar en la lista de objetos
input long              InpZOrder=0;             // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el signo "Mal"                                              |
//+------------------------------------------------------------------+
bool ArrowThumbDownCreate(const long              chart_ID=0,           // ID del grái
                          const string            name="ThumbDown",     // nombre del
                          const int               sub_window=0,         // número de s
                          datetime                time=0,               // hora del pu
                          double                  price=0,              // precio del
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // modo de anc
                          const color             clr=clrRed,           // color del s
                          const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo de l
                          const int               width=3,              // tamaño del
                          const bool              back=false,           // al fondo
                          const bool              selection=true,       // seleccionar
                          const bool              hidden=true,          // ocultar en
                          const long              z_order=0)            // prioridad p
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el signo
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el signo \"Mal\"! Código del error = ",GetLastError());
      return(false);
     }
//--- establecemos el modo de anclaje
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- establecemos el color del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea del contorno
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento del signo con rató
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool ArrowThumbDownMove(const long   chart_ID=0,        // ID del gráfico
                        const string name="ThumbDown", // nombre del objeto
                        datetime     time=0,           // coordenada del tiempo del pu
                        double       price=0)          // coordenada del precio del pu
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el modo de anclaje del signo "Mal"                        |
//+------------------------------------------------------------------+
bool ArrowThumbDownAnchorChange(const long              chart_ID=0,         // ID del g
                                const string            name="ThumbDown",   // nombre d
                                const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // modo de
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el modo de anclaje
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el modo de anclaje! Código del error = ",GetLastErro
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el signo "Mal"                                           |
//+------------------------------------------------------------------+
bool ArrowThumbDownDelete(const long   chart_ID=0,       // ID del gráfico
                          const string name="ThumbDown") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el signo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el signo \"Mal\"! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
```

© 2000-2025, MetaQuotes Ltd.

```
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas del punto de anclaje del signo
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el signo
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- creamos el signo "Mal" en el gráfico
   if(!ArrowThumbDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el punto de anclaje y cambiar su posición respecto al signo
//--- contador del ciclo
   int h_steps=bars/4;
//--- movemos el punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d<bars-1)
         d+=1;
      //--- movemos el punto
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
       //--- redibujamos el gráfico
       ChartRedraw();
       // retardo de 0,05 segundo
       Sleep(50);
      }
//--- retardo de 1 segundo
   Sleep(1000);
//--- contador del ciclo
   int v_steps=accuracy/4;
//--- movemos el punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p>1)
         p-=1;
      //--- movemos el punto
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- cambiamos la posición del punto de anclaje respecto al signo
   ArrowThumbDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- redibujamos el gráfico
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el signo desde el gráfico
   ArrowThumbDownDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_UP

Signo "Flecha arriba".



Nota

Desde la enumeración ENUM_ARROW_ANCHOR se puede elegir la posición del punto de anclaje respecto al signo.

Los signos de gran tamaño (más de 5) se puede crear sólo poniendo el correspondiente valor de la propiedad OBJPROP_WIDTH durante la escritura del código en MetaEditor.

Ejemplo

El siguiente script crea y desplaza el signo "Flecha arriba" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El scribt dibuja el signo \"Flecha arriba\"."
#property description "La coordenada del punto de anclaje se establece en"
#property description "por cientos de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="ArrowUp";     // Nombre del signo
input int               InpDate=25;            // Fecha del punto de anclaje en %
input int               InpPrice=25;           // Precio del punto de anclaje en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // Modo de anclaje
input color             InpColor=clrRed;       // Color del signo
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Estilo de la línea del contorno
input int               InpWidth=5;           // Tamaño del signo
input bool              InpBack=false;        // Signo al fondo
input bool              InpSelection=false;   // Seleccionar para mover
input bool              InpHidden=true;       // Ocultar en la lista de objetos
input long              InpZOrder=0;          // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el signo "Flecha arriba"                                    |
//+------------------------------------------------------------------+
bool ArrowUpCreate(const long              chart_ID=0,           // ID del gráfico
                   const string            name="ArrowUp",       // nombre del objeto
                   const int               sub_window=0,         // número de subventa
                   datetime                time=0,               // hora del punto de
                   double                  price=0,              // precio del punto d
                   const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // modo de anclaje
                   const color             clr=clrRed,           // color del signo
                   const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo de la línea
                   const int               width=3,              // tamaño del signo
                   const bool              back=false,           // al fondo
                   const bool              selection=true,       // seleccionar para m
                   const bool              hidden=true,          // ocultar en la list
                   const long              z_order=0)            // prioridad para el
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el signo
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el signo \"Flecha arriba\"! Código del error = ",GetLas
      return(false);
     }
//--- establecemos el modo de anclaje
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- establecemos el color del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea del contorno
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento del signo con rató
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool ArrowUpMove(const long   chart_ID=0,     // ID del gráfico
                 const string name="ArrowUp", // nombre del objeto
                 datetime     time=0,         // coordenada del tiempo del punto de an
                 double       price=0)        // coordenada del precio del punto de an
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el modo de anclaje del signo "Flecha arriba"              |
//+------------------------------------------------------------------+
bool ArrowUpAnchorChange(const long              chart_ID=0,        // ID del gráfico
                         const string            name="ArrowUp",    // nombre del obje
                         const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // modo de anclaje
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos la posición del punto de anclaje
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el modo de anclaje! Código del error = ",GetLastErro
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el signo "Flecha arriba"                                 |
//+------------------------------------------------------------------+
bool ArrowUpDelete(const long   chart_ID=0,     // ID del gráfico
                   const string name="ArrowUp") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el signo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el signo \"Flecha arriba\"! Código del error = ",Get
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas del punto de anclaje del signo
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el signo
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- creamos el signo "Flecha arriba" en el gráfico
   if(!ArrowUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el punto de anclaje y cambiar su posición respecto al signo
//--- contador del ciclo
   int v_steps=accuracy/2;
//--- movemos el punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p<accuracy-1)
         p+=1;
      //--- movemos el punto
      if(!ArrowUpMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- cambiamos la posición del punto de anclaje respecto al signo
   ArrowUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redibujamos el gráfico
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el signo desde el gráfico
   ArrowUpDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_DOWN

Signo "Flecha abajo".



Nota

Desde la enumeración ENUM_ARROW_ANCHOR se puede elegir la posición del punto de anclaje respecto al signo.

Los signos de gran tamaño (más de 5) se puede crear sólo poniendo el correspondiente valor de la propiedad OBJPROP_WIDTH durante la escritura del código en MetaEditor.

Ejemplo

El siguiente script crea y desplaza el signo "Flecha abajo" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El scribt dibuja el signo \"Flecha abajo\"."
#property description "La coordenada del punto de anclaje se establece en"
#property description "por cientos de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="ArrowDown";      // Nombre del signo
input int               InpDate=75;               // Fecha del punto de anclaje en %
input int               InpPrice=75;              // Precio del punto de anclaje en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Modo de anclaje
input color             InpColor=clrRed;          // Color del signo
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;       // Estilo de la línea del contorno
```

© 2000-2025, MetaQuotes Ltd.

```
input int                InpWidth=5;              // Tamaño del signo
input bool               InpBack=false;           // Signo al fondo
input bool               InpSelection=false;      // Seleccionar para mover
input bool               InpHidden=true;          // Ocultar en la lista de objetos
input long               InpZOrder=0;             // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el signo "Flecha abajo"                                     |
//+------------------------------------------------------------------+
bool ArrowDownCreate(const long              chart_ID=0,           // ID del gráfico
                     const string            name="ArrowDown",     // nombre del signo
                     const int               sub_window=0,         // número de subver
                     datetime                time=0,               // hora del punto d
                     double                  price=0,              // precio del punto
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // modo de anclaje
                     const color             clr=clrRed,           // color del signo
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo de la lín
                     const int               width=3,              // tamaño del siglo
                     const bool              back=false,           // al fondo
                     const bool              selection=true,       // seleccionar para
                     const bool              hidden=true,          // ocultar en la li
                     const long              z_order=0)            // prioridad para e
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el signo
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el signo \"Flecha abajo\"! Código del error = ",GetLast
      return(false);
     }
//--- modo de anclaje
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- establecemos el color del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea del contorno
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento del signo con rató
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool ArrowDownMove(const long   chart_ID=0,        // ID del gráfico
                   const string name="ArrowDown", // nombre del objeto
                   datetime     time=0,           // coordenada del tiempo del punto d
                   double       price=0)          // coordenada del precio del punto d
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el modo de anclaje del signo "Flecha abajo"               |
//+------------------------------------------------------------------+
bool ArrowDownAnchorChange(const long              chart_ID=0,        // ID del gráfic
                           const string            name="ArrowDown",  // nombre del ob
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // modo de ancla
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos la posición del punto de anclaje
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el modo de anclaje! Código del error = ",GetLastErro
      return(false);
     }
//--- ejecución con éxito
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el signo "Flecha abajo"                                  |
//+------------------------------------------------------------------+
bool ArrowDownDelete(const long   chart_ID=0,       // ID del gráfico
                     const string name="ArrowDown") // nombre del signo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el signo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el signo \"Flecha abajo\"! Código del error = ",GetL
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
```

© 2000-2025, MetaQuotes Ltd.

```
//--- para establecer y modificar las coordenadas del punto de anclaje del signo
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el signo
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- creamos el signo "Flecha abajo" en el gráfico
   if(!ArrowDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el punto de anclaje y cambiar su posición respecto al signo
//--- contador del ciclo
   int v_steps=accuracy/2;
//--- movemos el punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p>1)
         p-=1;
      //--- movemos el punto
      if(!ArrowDownMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- cambiamos la posición del punto de anclaje respecto al signo
   ArrowDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- redibujamos el gráfico
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el signo desde el gráfico
   ArrowDownDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_STOP

Signo "Stop".



Nota

Desde la enumeración ENUM_ARROW_ANCHOR se puede elegir la posición del punto de anclaje respecto al signo.

Los signos de gran tamaño (más de 5) se puede crear sólo poniendo el correspondiente valor de la propiedad OBJPROP_WIDTH durante la escritura del código en MetaEditor.

Ejemplo

El siguiente script crea y desplaza el signo "Stop" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El scribt dibuja el signo \"Stop\"."
#property description "La coordenada del punto de anclaje se establece en"
#property description "por cientos de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="ArrowStop";      // Nombre del signo
input int               InpDate=10;               // Fecha del punto de anclaje en %
input int               InpPrice=50;              // Precio del punto de anclaje en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Modo de anclaje
input color             InpColor=clrRed;          // Color del signo
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Estilo de la línea del contorno
input int               InpWidth=5;              // Tamaño del signo
input bool              InpBack=false;           // Signo al fondo
input bool              InpSelection=false;      // Seleccionar para mover
input bool              InpHidden=true;          // Ocultar en la lista de objetos
input long              InpZOrder=0;             // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el signo "Stop"                                             |
//+------------------------------------------------------------------+
bool ArrowStopCreate(const long              chart_ID=0,           // ID del gráfico
                     const string            name="ArrowStop",     // nombre del signo
                     const int               sub_window=0,         // número de subver
                     datetime                time=0,               // hora del punto d
                     double                  price=0,              // precio del punto
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // modo de anclaje
                     const color             clr=clrRed,           // color del signo
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo de la lín
                     const int               width=3,              // tamaño del siglo
                     const bool              back=false,           // al fondo
                     const bool              selection=true,       // seleccionar para
                     const bool              hidden=true,          // ocultar en la li
                     const long              z_order=0)            // prioridad para e
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el signo
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_STOP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el signo \"Stop\"! Código del error = ",GetLastError())
      return(false);
     }
//--- establecemos el modo de anclaje
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- establecemos el color del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea del contorno
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento del signo con rató
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool ArrowStopMove(const long   chart_ID=0,        // ID del gráfico
                   const string name="ArrowStop", // nombre del objeto
                   datetime     time=0,           // coordenada del tiempo del punto d
                   double       price=0)          // coordenada del precio del punto d
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el modo de anclaje del signo "Stop"                       |
//+------------------------------------------------------------------+
bool ArrowStopAnchorChange(const long              chart_ID=0,        // ID del gráfic
                           const string            name="ArrowStop",  // nombre del ob
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // posición del
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el modo de anclaje
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el modo de anclaje! Código del error = ",GetLastErro
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el signo "Stop"                                          |
//+------------------------------------------------------------------+
bool ArrowStopDelete(const long   chart_ID=0,       // ID del gráfico
                     const string name="ArrowStop") // nombre del signo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el signo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el signo \"Stop\"! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas del punto de anclaje del signo
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el signo
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- creamos el signo "Stop" en el gráfico
   if(!ArrowStopCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el punto de anclaje y cambiar su posición respecto al signo
//--- contador del ciclo
   int h_steps=bars*2/5;
//--- movemos el punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d<bars-1)
         d+=1;
      //--- movemos el punto
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,025 segundo
      Sleep(25);
     }
//--- cambiamos la posición del punto de anclaje respecto al signo
   ArrowStopAnchorChange(0,InpName,ANCHOR_TOP);
//--- redibujamos el gráfico
   ChartRedraw();
//--- contador del ciclo
   h_steps=bars*2/5;
//--- movemos el punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d<bars-1)
         d+=1;
      //--- movemos el punto
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,025 segundo
      Sleep(25);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el signo desde el gráfico
   ArrowStopDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_CHECK

Signo "Marca".



Nota

Desde la enumeración ENUM_ARROW_ANCHOR se puede elegir la posición del punto de anclaje respecto al signo.

Los signos de gran tamaño (más de 5) se puede crear sólo poniendo el correspondiente valor de la propiedad OBJPROP_WIDTH durante la escritura del código en MetaEditor.

Ejemplo

El siguiente script crea y desplaza el signo "Marca" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El scribt dibuja el signo \"Marca\"."
#property description "La coordenada del punto de anclaje se establece en"
#property description "por cientos de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="ArrowCheck"; // Nombre del signo
input int               InpDate=10;           // Fecha del punto de anclaje en %
input int               InpPrice=50;          // Precio del punto de anclaje en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // Modo de anclaje
input color             InpColor=clrRed;      // Color del signo
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Estilo de la línea del contorno
input int               InpWidth=5;           // Tamaño del signo
input bool              InpBack=false;        // Signo al fondo
input bool              InpSelection=false;   // Seleccionar para mover
input bool              InpHidden=true;       // Ocultar en la lista de objetos
input long              InpZOrder=0;          // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el signo "Marca"                                            |
//+------------------------------------------------------------------+
bool ArrowCheckCreate(const long              chart_ID=0,           // ID del gráfico
                      const string            name="ArrowCheck",    // nombre del sign
                      const int               sub_window=0,         // número de subve
                      datetime                time=0,               // hora del punto
                      double                  price=0,              // precio del punt
                      const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // modo de anclaje
                      const color             clr=clrRed,           // color del signo
                      const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo de la li
                      const int               width=3,              // tamaño del sign
                      const bool              back=false,           // al fondo
                      const bool              selection=true,       // seleccionar par
                      const bool              hidden=true,          // ocultar en la l
                      const long              z_order=0)            // prioridad para
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el signo
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_CHECK,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el signo \"Marca\"! Código del error = ",GetLastError()
      return(false);
     }
//--- establecemos el modo de anclaje
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- establecemos el color del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea del contorno
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento del signo con rató
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear con el ratón sobre
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool ArrowCheckMove(const long   chart_ID=0,        // ID del gráfico
                    const string name="ArrowCheck", // nombre del objeto
                    datetime     time=0,            // coordenada del tiempo del punto
                    double       price=0)           // coordenada del precio del punto
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el modo de anclaje del signo "Marca"                      |
//+------------------------------------------------------------------+
bool ArrowCheckAnchorChange(const long              chart_ID=0,        // ID del gráfi
                            const string            name="ArrowCheck", // nombre del o
                            const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // modo de ancl
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el modo de anclaje
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el modo de anclaje! Código del error = ",GetLastErro
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el signo "Marca"                                         |
//+------------------------------------------------------------------+
bool ArrowCheckDelete(const long   chart_ID=0,        // ID del gráfico
                      const string name="ArrowCheck") // nombre del signo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el signo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el signo \"Marca\"! Código del error = ",GetLastErro
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas del punto de anclaje del signo
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar el signo
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- creamos el signo "Marca" en el gráfico
   if(!ArrowCheckCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el punto de anclaje y cambiar su posición respecto al signo
//--- contador del ciclo
   int h_steps=bars*2/5;
//--- movemos el punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d<bars-1)
         d+=1;
      //--- movemos el punto
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,025 segundo
      Sleep(25);
     }
//--- cambiamos la posición del punto de anclaje respecto al signo
   ArrowCheckAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redibujamos el gráfico
   ChartRedraw();
//--- contador del ciclo
   h_steps=bars*2/5;
//--- movemos el punto de anclaje
   for(int i=0;i<h_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(d<bars-1)
         d+=1;
      //--- movemos el punto
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,025 segundo
      Sleep(25);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el signo desde el gráfico
   ArrowCheckDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_LEFT_PRICE

Etiqueta izquierda de precio.



Ejemplo

El siguiente script crea y desplaza la etiqueta izquierda de precio en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea la etiqueta izquierda de precio en el gráfico."
#property description "La coordenada del punto de anclaje se establece en por cientos"
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="LeftPrice";  // Nombre de la etiqueta de precio
input int               InpDate=100;          // Fecha del punto de anclaje en %
input int               InpPrice=10;          // Precio del punto de anclaje en %
input color             InpColor=clrRed;      // color de la etiqueta de precio
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Estilo de la línea del contorno
input int               InpWidth=2;           // Tamaño de la etiqueta de precio
input bool              InpBack=false;        // Etiqueta al fondo
input bool              InpSelection=true;    // Seleccionar para mover
input bool              InpHidden=true;       // Ocultar en la lista de objetos
input long              InpZOrder=0;          // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea la etiqueta izquierda de precio                             |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ArrowLeftPriceCreate(const long            chart_ID=0,        // ID del gráfico
                          const string          name="LeftPrice",  // nombre de la eti
                          const int             sub_window=0,      // número de subver
                          datetime              time=0,            // hora del punto d
                          double                price=0,           // precio del punto
                          const color           clr=clrRed,        // color de la etic
                          const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de la lín
                          const int             width=1,           // tamaño de la eti
                          const bool            back=false,        // al fondo
                          const bool            selection=true,    // seleccionar para
                          const bool            hidden=true,       // ocultar en la li
                          const long            z_order=0)         // prioridad para e
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la etiqueta de precio
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_LEFT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la etiqueta izquierda de precio! Código del error = ",G
      return(false);
     }
//--- establecemos el color de la etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea del contorno
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño de la etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la etiqueta cor
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ArrowLeftPriceMove(const long   chart_ID=0,        // ID del gráfico
                        const string name="LeftPrice", // nombre de la etiqueta
                        datetime     time=0,           // coordenada del tiempo del pu
                        double       price=0)          // coordenada del precio del pu
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina la etiqueta izquierda de precio desde el gráfico
//+------------------------------------------------------------------+
bool ArrowLeftPriceDelete(const long   chart_ID=0,       // ID del gráfico
                          const string name="LeftPrice") // nombre de la etiqueta
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la etiqueta izquierda de precio! Código del error =
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas del punto de anclaje de la etiqueta
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError());
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar la etiqueta
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- creamos la etiqueta izquierda de precio en el gráfico
   if(!ArrowLeftPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el punto de anclaje
//--- contador del ciclo
   int v_steps=accuracy*4/5;
//--- movemos el punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p<accuracy-1)
         p+=1;
      //--- movemos el punto
      if(!ArrowLeftPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos la etiqueta desde el gráfico
   ArrowLeftPriceDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_RIGHT_PRICE

Etiqueta derecha de precio.



Ejemplo

El siguiente script crea y desplaza la etiqueta derecha de precio en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea la etiqueta derecha de precio en el gráfico."
#property description "La coordenada del punto de anclaje se establece en por cientos"
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="RightPrice"; // Nombre de la etiqueta de precio
input int               InpDate=0;            // Fecha del punto de anclaje en %
input int               InpPrice=90;          // Precio del punto de anclaje en %
input color             InpColor=clrRed;      // color de la etiqueta de precio
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Estilo de la línea del contorno
input int               InpWidth=2;           // Tamaño de la etiqueta de precio
input bool              InpBack=false;        // Etiqueta al fondo
input bool              InpSelection=true;    // Seleccionar para mover
input bool              InpHidden=true;       // Ocultar en la lista de objetos
input long              InpZOrder=0;          // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea la etiqueta derecha de precio                               |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ArrowRightPriceCreate(const long            chart_ID=0,        // ID del gráfico
                           const string          name="RightPrice", // nombre de la et
                           const int             sub_window=0,      // número de subve
                           datetime              time=0,            // hora del punto
                           double                price=0,           // precio del punt
                           const color           clr=clrRed,        // color de la eti
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de la lí
                           const int             width=1,           // tamaño de la et
                           const bool            back=false,        // al fondo
                           const bool            selection=true,    // seleccionar par
                           const bool            hidden=true,       // ocultar en la l
                           const long            z_order=0)         // prioridad para
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la etiqueta de precio
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_RIGHT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la etiqueta derecha de precio! Código del error = ",Get
      return(false);
     }
//--- establecemos el color de la etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea del contorno
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño de la etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la etiqueta cor
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ArrowRightPriceMove(const long   chart_ID=0,        // ID del gráfico
                         const string name="RightPrice", // nombre de la etiqueta
                         datetime     time=0,            // coordenada del tiempo del
                         double       price=0)           // coordenada del precio del
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina la etiqueta derecha de precio desde el gráfico
//+------------------------------------------------------------------+
bool ArrowRightPriceDelete(const long   chart_ID=0,        // ID del gráfico
                           const string name="RightPrice") // nombre de la etiqueta
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la etiqueta derecha de precio! Código del error = ",
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas del punto de anclaje de la etiqueta
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar la etiqueta
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- creamos la etiqueta derecha de precio en el gráfico
   if(!ArrowRightPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

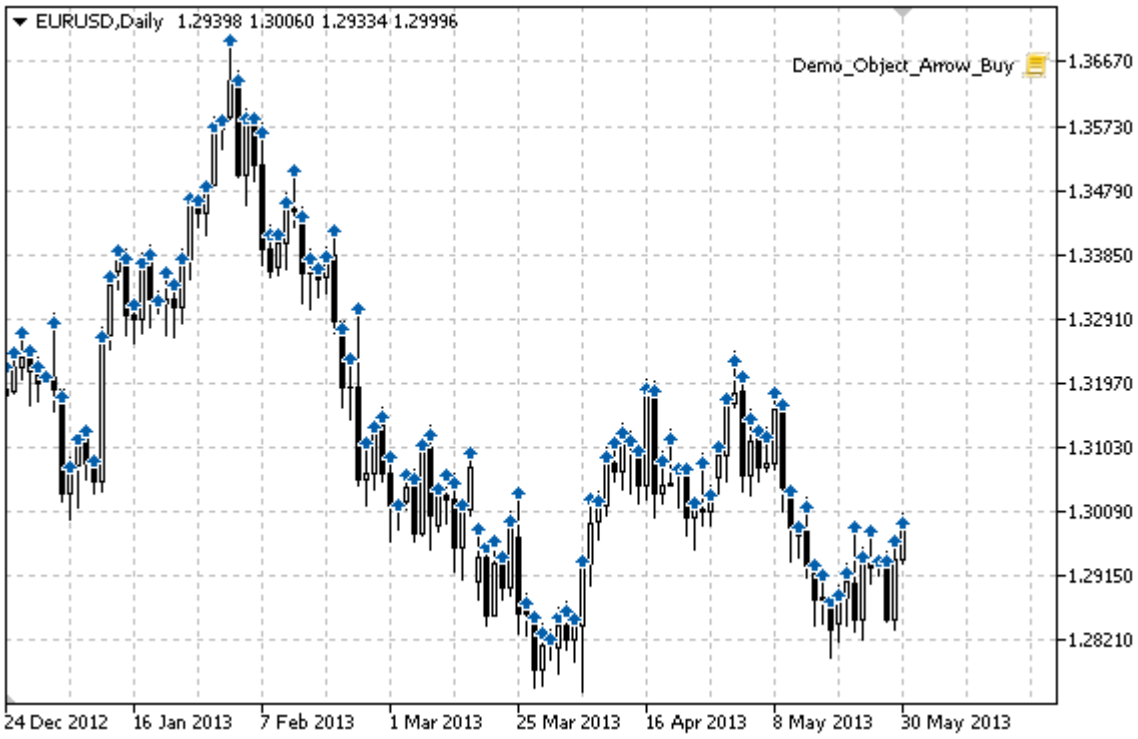
© 2000-2025, MetaQuotes Ltd.

```
      }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el punto de anclaje
//--- contador del ciclo
   int v_steps=accuracy*4/5;
//--- movemos el punto de anclaje
   for(int i=0;i<v_steps;i++)
     {
      //--- cogemos el siguiente valor
      if(p>1)
         p-=1;
      //--- movemos el punto
      if(!ArrowRightPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos la etiqueta desde el gráfico
   ArrowRightPriceDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_BUY

Signo "Buy".



Ejemplo

El siguiente script crea y desplaza el signo "Buy" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script dibuja el signo \"Buy\" en la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input color InpColor=C'3,95,172'; // Color de los signos
//+------------------------------------------------------------------+
//| Crea el signo "Buy"                                              |
//+------------------------------------------------------------------+
bool ArrowBuyCreate(const long            chart_ID=0,        // ID del gráfico
                    const string          name="ArrowBuy",   // nombre del signo
                    const int             sub_window=0,      // número de subventana
                    datetime              time=0,            // hora del punto de anc
                    double                price=0,           // precio del punto de ar
                    const color           clr=C'3,95,172',   // color del signo
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de la línea (du
                    const int             width=1,           // tamaño de la línea (du
                    const bool            back=false,        // al fondo
                    const bool            selection=false,   // seleccionar para mover
```

© 2000-2025, MetaQuotes Ltd.

```
                    const bool            hidden=true,       // ocultar en la lista de
                    const long            z_order=0)         // prioridad para el clic
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el signo
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_BUY,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el signo \"Buy\"! Código del error = ",GetLastError());
      return(false);
     }
//--- establecemos el color del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea (durante la selección)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño de la línea (durante la selección)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento del signo con ratón
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear con el ratón sobre
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool ArrowBuyMove(const long   chart_ID=0,       // ID del gráfico
                  const string name="ArrowBuy", // nombre del objeto
                  datetime     time=0,          // coordenada del tiempo del punto de
                  double       price=0)         // coordenada del precio del punto de
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el signo "Buy"                                           |
//+------------------------------------------------------------------+
bool ArrowBuyDelete(const long   chart_ID=0,      // ID del gráfico
                    const string name="ArrowBuy") // nombre del signo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el signo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el signo \"Buy\"! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // array para guardar las fechas de las barras visibles
   double   low[];  // array para guardar los precios Low de las barras visibles
   double   high[]; // array para guardar los precios High de las barras visibles
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("¡Fallo al copiar los valores de los precios Low! Código del error = ",Get
      return;
     }
//--- llenamos el array de precios High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("¡Fallo al copiar los valores de los precios High! Código del error = ",Ge
      return;
     }
//--- creamos los signos "Buy" en el punto Low para barra visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyCreate(0,"ArrowBuy_"+(string)i,0,date[i],low[i],InpColor))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- movemos los signos "Buy" al punto High para cada barra visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyMove(0,"ArrowBuy_"+(string)i,date[i],high[i]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- eliminamos los signos "Buy"
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyDelete(0,"ArrowBuy_"+(string)i))
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_SELL

Signo "Sell".



Ejemplo

El siguiente script crea y desplaza el signo "Sell" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script dibuja el signo \"Sell\" en la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input color InpColor=C'225,68,29'; // Color de signos
//+------------------------------------------------------------------+
//| Crea el signo "Sell"                                             |
//+------------------------------------------------------------------+
bool ArrowSellCreate(const long            chart_ID=0,        // ID del gráfico
                     const string          name="ArrowSell",  // nombre del signo
                     const int             sub_window=0,      // número de subventana
                     datetime              time=0,            // hora del punto de anc
                     double                price=0,           // precio del punto de a
                     const color           clr=C'225,68,29',  // color del signo
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de la línea (c
                     const int             width=1,           // tamaño de la línea (c
                     const bool            back=false,        // al fondo
                     const bool            selection=false,   // seleccionar para move
```

© 2000-2025, MetaQuotes Ltd.

```
                      const bool            hidden=true,       // ocultar en la lista d
                      const long            z_order=0)         // prioridad para el cli
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el signo
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_SELL,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el signo \"Sell\"! Código del error = ",GetLastError())
      return(false);
     }
//--- establecemos el color del signo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea (durante la selección)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño de la línea (durante la selección)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento del signo con ratón
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool ArrowSellMove(const long   chart_ID=0,       // ID del gráfico
                   const string name="ArrowSell", // nombre del objeto
                   datetime     time=0,           // coordenada del tiempo del punto d
                   double       price=0)          // coordenada del precio del punto d
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el signo "Sell"                                          |
//+------------------------------------------------------------------+
bool ArrowSellDelete(const long   chart_ID=0,       // ID del gráfico
                     const string name="ArrowSell") // nombre del gráfico
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el signo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el signo \"Sell\"! Código del error = ",GetLastErro
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // array para guardar las fechas de las barras visibles
   double   low[];  // array para guardar los precios Low de las barras visibles
   double   high[]; // array para guardar los precios High de las barras visibles
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("¡Fallo al copiar los valores de los precios Low! Código del error = ",Get
      return;
     }
//--- llenamos el array de precios High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("¡Fallo al copiar los valores de los precios High! Código del error = ",Ge
      return;
     }
//--- creamos los signos "Sell" en el punto High para barra visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellCreate(0,"ArrowSell_"+(string)i,0,date[i],high[i],InpColor))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- movemos los signos "Sell" al punto Low para cada barra visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellMove(0,"ArrowSell_"+(string)i,date[i],low[i]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- eliminamos los signos "Sell"
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellDelete(0,"ArrowSell_"+(string)i))
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW

Objeto "Flecha".



Nota

Desde la enumeración ENUM_ARROW_ANCHOR se puede elegir la posición del punto de anclaje respecto al signo.

Las flechas de gran tamaño (más de 5) se puede crear sólo poniendo el correspondiente valor de la propiedad OBJPROP_WIDTH durante la escritura del código en MetaEditor.

Se puede elegir el tipo necesario de la flecha poniendo uno de los códigos del símbolo de la fuente Wingdings.

Ejemplo

El siguiente script crea el objeto "Flecha" en el gráfico y cambia su tipo. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea la flecha aleatoria en la ventana del gráfico."
#property description "La coordenada del punto de anclaje se establece en por cientos'
#property description "de las dimensiones de la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="Arrow";        // Nombre de la flecha
input int               InpDate=50;             // Fecha del punto de anclaje en %
```

© 2000-2025, MetaQuotes Ltd.

```
input int                InpPrice=50;             // Precio del punto de anclaje en %
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;    // Modo de anclaje
input color              InpColor=clrDodgerBlue;  // Color de la flecha
input ENUM_LINE_STYLE    InpStyle=STYLE_SOLID;    // Estilo de la línea del contorno
input int                InpWidth=10;             // Tamaño de la flecha
input bool               InpBack=false;           // Flecha al fondo
input bool               InpSelection=false;      // Seleccionar para mover
input bool               InpHidden=true;          // Ocultar en la lista de objetos
input long               InpZOrder=0;             // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea la flecha                                                   |
//+------------------------------------------------------------------+
bool ArrowCreate(const long              chart_ID=0,           // ID del gráfico
                 const string            name="Arrow",         // nombre de la flecha
                 const int               sub_window=0,         // número de subventana
                 datetime                time=0,               // hora del punto de an
                 double                  price=0,              // precio del punto de
                 const uchar             arrow_code=252,       // código de la flecha
                 const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // posición del punto c
                 const color             clr=clrRed,           // color de la flecha
                 const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo de la línea c
                 const int               width=3,              // tamaño de la flecha
                 const bool              back=false,           // al fondo
                 const bool              selection=true,       // seleccionar para mov
                 const bool              hidden=true,          // ocultar en la lista
                 const long              z_order=0)            // prioridad para el cl
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeArrowEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la flecha
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la flecha! Código del error = ",GetLastError());
      return(false);
     }
//--- establecemos el código de la flecha
   ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,arrow_code);
//--- establecemos el modo de anclaje
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- establecemos el color de la flecha
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de la línea del contorno
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño de la flecha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la flecha con r
//--- cuando el objeto gráfico se crea usando la función ObjectCreate, por defecto el
//--- no se puede seleccionar y mover. Mientras que dentro de este método el parámetro
//--- por defecto es igual a true, lo que permite seleccionar y mover este objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool ArrowMove(const long   chart_ID=0,   // ID del gráfico
               const string name="Arrow", // nombre del objeto
               datetime     time=0,       // coordenada del tiempo del punto de anclaj
               double       price=0)      // coordenada del precio del punto de anclaj
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el código de la flecha                                    |
//+------------------------------------------------------------------+
bool ArrowCodeChange(const long   chart_ID=0,   // ID del gráfico
                     const string name="Arrow", // nombre del objeto
                     const uchar  code=252)     // código de la flecha
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el código de la flecha
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,code))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el código de la flecha! Código del error = ",GetLastE
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el modo de anclaje                                        |
//+------------------------------------------------------------------+
bool ArrowAnchorChange(const long              chart_ID=0,        // ID del gráfico
                       const string            name="Arrow",      // nombre del objeto
                       const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // modo del anclaje
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el modo de anclaje
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el modo de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina la flecha                                                |
//+------------------------------------------------------------------+
bool ArrowDelete(const long   chart_ID=0,   // ID del gráfico
                 const string name="Arrow") // nombre de la flecha
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la flecha
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la flecha! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamaño del array price
   int accuracy=1000;
//--- arrays para guardar los valores de las fechas y precios que van a utilizarse
//--- para establecer y modificar las coordenadas del punto de anclaje del signo
   datetime date[];
   double   price[];
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios
//--- encontramos el valor máximo y mínimo del gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- determinamos el paso del cambio del precio y llenamos el array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definimos los puntos para trazar la flecha
   int d=InpDate*(bars-1)/100;
```

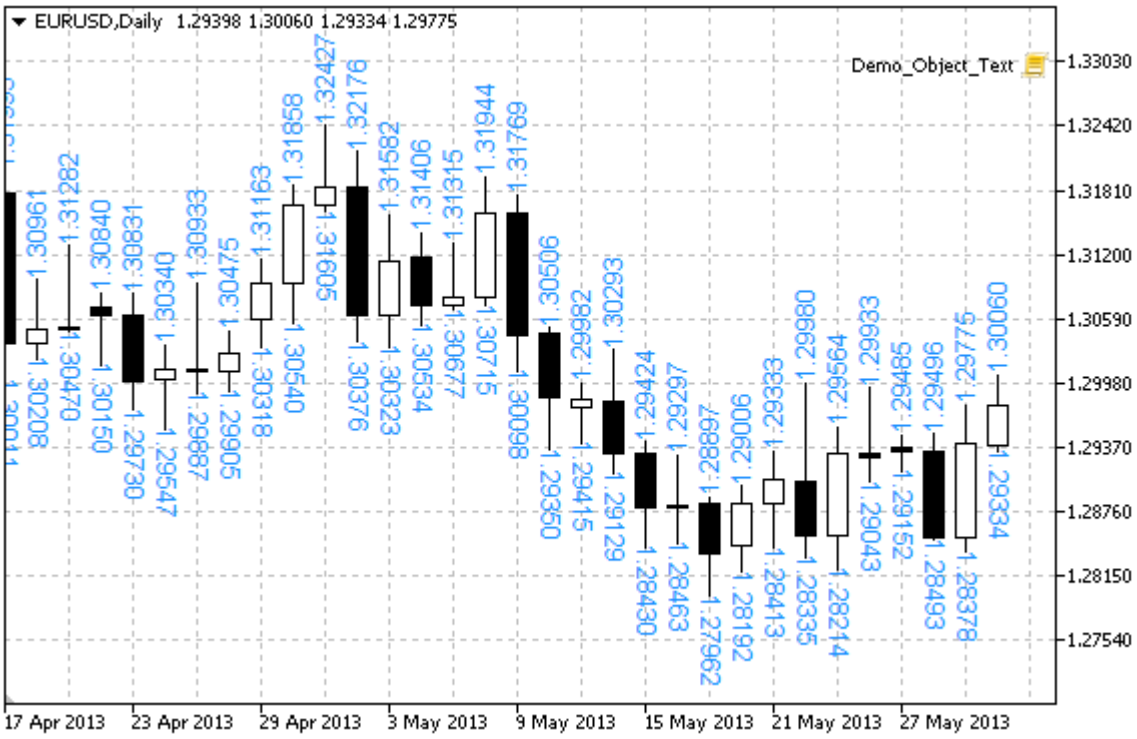
© 2000-2025, MetaQuotes Ltd.

```
   int p=InpPrice*(accuracy-1)/100;
//--- creamos la flecha en el gráfico
   if(!ArrowCreate(0,InpName,0,date[d],price[p],32,InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico
   ChartRedraw();
//--- vamos a ver en el ciclo todas las variantes de creación de las flechas
   for(int i=33;i<256;i++)
     {
      if(!ArrowCodeChange(0,InpName,(uchar)i))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de medio segundo
      Sleep(500);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos la flecha desde el gráfico
   ArrowDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TEXT

Objeto "Texto".



Nota

Desde la enumeración ENUM_ANCHOR_POINT se puede elegir la posición del punto de anclaje respecto al texto. Además, se puede cambiar el ángulo de inclinación del texto a través de la propiedad OBJPROP_ANGLE.

Ejemplo

El siguiente script crea varios objetos "Texto" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea el objeto gráfico \"Texto\"."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string             InpFont="Arial";          // Fuente
input int                InpFontSize=10;           // Tamaño de la fuente
input color              InpColor=clrRed;          // Color
input double             InpAngle=90.0;            // Ángulo de inclinación en grados
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_LEFT;    // Modo de anclaje
input bool               InpBack=false;            // Objeto al fondo
input bool               InpSelection=false;       // Seleccionar para mover
input bool               InpHidden=true;           // Ocultar en la lista de objetos
```

© 2000-2025, MetaQuotes Ltd.

```
input long              InpZOrder=0;             // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el objeto "Texto"                                           |
//+------------------------------------------------------------------+
bool TextCreate(const long              chart_ID=0,               // ID del gráfico
                const string            name="Text",              // nombre del objeto
                const int               sub_window=0,             // número de subvent
                datetime                time=0,                   // hora del punto de
                double                  price=0,                  // precio del punto
                const string            text="Text",              // el texto
                const string            font="Arial",             // fuente
                const int               font_size=10,             // tamaño de la fuer
                const color             clr=clrRed,               // color
                const double            angle=0.0,                // inclinación del t
                const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // modo de anclaje
                const bool              back=false,               // al fondo
                const bool              selection=false,          // seleccionar para
                const bool              hidden=true,              // ocultar en la lis
                const long              z_order=0)                // prioridad para el
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeTextEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el objeto "Texto"
   if(!ObjectCreate(chart_ID,name,OBJ_TEXT,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el objeto \"Texto\"! Código del error = ",GetLastError
      return(false);
     }
//--- ponemos el texto
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- establecemos la fuente del texto
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- establecemos el tamaño del texto
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- establecemos el ángulo de inclinación del texto
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- establecemos el modo de anclaje
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- establecemos el color
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento del texto con rató
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el punto de anclaje                                        |
//+------------------------------------------------------------------+
bool TextMove(const long   chart_ID=0,  // ID del gráfico
              const string name="Text", // nombre del objeto
              datetime     time=0,      // coordenada del tiempo del punto de anclaje
              double       price=0)     // coordenada del precio del punto de anclaje
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el texto del objeto                                       |
//+------------------------------------------------------------------+
bool TextChange(const long   chart_ID=0,  // ID del gráfico
                const string name="Text", // nombre del objeto
                const string text="Text") // texto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el texto del objeto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el texto! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Elimina el objeto "Texto"                                        |
//+------------------------------------------------------------------+
bool TextDelete(const long   chart_ID=0,  // ID del gráfico
                const string name="Text") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el objeto \"Texto\"! Código del error = ",GetLastEr
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeTextEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // array para guardar las fechas de las barras visibles
   double   low[];  // array para guardar los precios Low de las barras visibles
   double   high[]; // array para guardar los precios High de las barras visibles
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
       return;
      }
//--- llenamos el array de precios Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
      {
       Print("¡Fallo al copiar los valores de los precios Low! Código del error = ",Get
       return;
      }
//--- llenamos el array de precios High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
      {
       Print("¡Fallo al copiar los valores de los precios High! Código del error = ",Ge
       return;
      }
//--- definimos con qué frecuencia hay que hacer las notas
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- definimos el paso
   int step=1;
   switch(scale)
      {
       case 0:
          step=12;
          break;
       case 1:
          step=6;
          break;
       case 2:
          step=4;
          break;
       case 3:
          step=2;
          break;
      }
//--- creamos las notas para los valores High y Low de las barras (con intervalos)
   for(int i=0;i<bars;i+=step)
      {
       //--- creamos las notas
       if(!TextCreate(0,"TextHigh_"+(string)i,0,date[i],high[i],DoubleToString(high[i],
          InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
       if(!TextCreate(0,"TextLow_"+(string)i,0,date[i],low[i],DoubleToString(low[i],5),
          InpColor,-InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de medio segundo
   Sleep(500);
//--- eliminamos las notas
   for(int i=0;i<bars;i+=step)
     {
      if(!TextDelete(0,"TextHigh_"+(string)i))
         return;
      if(!TextDelete(0,"TextLow_"+(string)i))
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_LABEL

Objeto "Etiqueta de texto".



Nota

Desde la enumeración ENUM_ANCHOR_POINT se puede elegir la posición del punto de anclaje respecto a la etiqueta. Las coordenadas del punto de anclaje se establecen en píxeles.

Además, desde la enumeración ENUM_BASE_CORNER se puede elegir la esquina de enlace de la etiqueta de texto.

Ejemplo

El siguiente script crea y desplaza la "Etiqueta de texto" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea el objeto gráfico \"Etiqueta de texto\"."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="Label";          // Nombre de la etiqueta
input int               InpX=150;                 // Distancia por el eje X
input int               InpY=150;                 // Distancia por el eje Y
input string            InpFont="Arial";          // Fuente
input int               InpFontSize=14;           // Tamaño de la fuente
input color             InpColor=clrRed;          // Color
input double            InpAngle=0.0;             // Ángulo de inclinación en grados
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // Modo de anclaje
input bool              InpBack=false;           // Objeto al fondo
input bool              InpSelection=true;       // Seleccionar para mover
input bool              InpHidden=true;          // Ocultar en la lista de objetos
input long              InpZOrder=0;             // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea la etiqueta de texto                                        |
//+------------------------------------------------------------------+
bool LabelCreate(const long              chart_ID=0,               // ID del gráfico
                 const string            name="Label",             // nombre de la eti
                 const int               sub_window=0,             // número de subver
                 const int               x=0,                      // coordenada por e
                 const int               y=0,                      // coordenada por e
                 const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // esquina del grái
                 const string            text="Label",             // texto
                 const string            font="Arial",             // fuente
                 const int               font_size=10,             // tamaño de la fue
                 const color             clr=clrRed,               // color
                 const double            angle=0.0,                // inclinación del
                 const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // modo de anclaje
                 const bool              back=false,               // al fondo
                 const bool              selection=false,          // seleccionar para
                 const bool              hidden=true,              // ocultar en la li
                 const long              z_order=0)                //prioridad para el
  {
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la etiqueta de texto
   if(!ObjectCreate(chart_ID,name,OBJ_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la etiqueta de texto! Código del error = ",GetLastError
      return(false);
     }
//--- establecemos las coordenadas de la etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- establecemos la esquina del gráfico respecto a la cual van a determinarse las co
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- ponemos el texto
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- establecemos la fuente del texto
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- establecemos el tamaño del texto
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- establecemos el ángulo de inclinación del texto
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- establecemos el modo de anclaje
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- establecemos el color
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la etiqueta cor
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve la etiqueta de texto                                       |
//+------------------------------------------------------------------+
bool LabelMove(const long   chart_ID=0,   // ID del gráfico
               const string name="Label", // nombre de la etiqueta
               const int    x=0,          // coordenada por el eje X
               const int    y=0)          // coordenada por el eje Y
  {
//--- anulamos el valor del error
   ResetLastError();
//--- movemos la etiqueta de texto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada X de la etiqueta! Código del error = ",Ge
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada Y de la etiqueta! Código del error = ",Ge
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia la esquina del gráfico para el enlace de la etiqueta      |
//+------------------------------------------------------------------+
bool LabelChangeCorner(const long             chart_ID=0,               // ID del grá
                       const string           name="Label",             // nombre de
                       const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // esquina del
  {
//--- anulamos el valor del error
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- cambiamos la esquina de anclaje
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la esquina de anclaje! Código del error = ",GetLastEr
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el texto del objeto                                       |
//+------------------------------------------------------------------+
bool LabelTextChange(const long   chart_ID=0,   // ID del gráfico
                     const string name="Label", // nombre del objeto
                     const string text="Text")  // texto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el texto del objeto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el texto! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina la etiqueta de texto                                     |
//+------------------------------------------------------------------+
bool LabelDelete(const long   chart_ID=0,   // ID del gráfico
                 const string name="Label") // nombre de la etiqueta
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar la etiqueta de texto! Código del error = ",GetLastEr
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- recordamos las coordenadas de la etiqueta en las variables locales
   int x=InpX;
   int y=InpY;
//--- tamaño de la ventana del gráfico
   long x_distance;
   long y_distance;
//--- definimos las dimensiones de la ventana
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("¡Fallo al obtener el ancho del gráfico! Código del error = ",GetLastError
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("¡Fallo al obtener el alto del gráfico! Código del error = ",GetLastError
      return;
     }
//--- comprobamos si los parámetros de entrada son correctos
   if(InpX<0 || InpX>x_distance-1 || InpY<0 || InpY>y_distance-1)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- preparamos el texto inicial para la etiqueta
   string text;
   StringConcatenate(text,"La esquina superior izquierda: ",x,",",y);
//--- creamos la etiqueta de texto en el gráfico
   if(!LabelCreate(0,InpName,0,InpX,InpY,CORNER_LEFT_UPPER,text,InpFont,InpFontSize,
      InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos medio segundo
   ChartRedraw();
   Sleep(500);
//--- vamos a mover la etiqueta y a la vez cambiar su texto
//--- número de iteraciones por ejes
   int h_steps=(int)(x_distance/2-InpX);
   int v_steps=(int)(y_distance/2-InpY);
//--- movemos la etiqueta abajo
   for(int i=0;i<v_steps;i++)
     {
      //--- cambiamos la coordenada
      y+=2;
      //--- movemos la etiqueta y cambiamos su texto
      MoveAndTextChange(x,y,"La esquina superior izquierda: ");
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- retardo de medio segundo
   Sleep(500);
//--- movemos la etiqueta a la derecha
   for(int i=0;i<h_steps;i++)
     {
      //--- cambiamos la coordenada
      x+=2;
      //--- movemos la etiqueta y cambiamos su texto
      MoveAndTextChange(x,y,"La esquina superior izquierda: ");
     }
//--- retardo de medio segundo
   Sleep(500);
//--- movemos la etiqueta para arriba
   for(int i=0;i<v_steps;i++)
     {
      //--- cambiamos la coordenada
      y-=2;
      //--- movemos la etiqueta y cambiamos su texto
      MoveAndTextChange(x,y,"La esquina superior izquierda: ");
     }
//--- retardo de medio segundo
   Sleep(500);
//--- movemos la etiqueta a la izquierda
   for(int i=0;i<h_steps;i++)
     {
      //--- cambiamos la coordenada
      x-=2;
      //--- movemos la etiqueta y cambiamos su texto
      MoveAndTextChange(x,y,"La esquina superior izquierda: ");
     }
//--- retardo de medio segundo
   Sleep(500);
//--- ahora movemos el punto mediante el cambio de la esquina de enlace
//--- movemos a la esquina inferior izquierda
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_LOWER))
      return;
//--- cambiamos el texto de la etiqueta
   StringConcatenate(text,"La esquina inferior izquierda: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redibujamos el gráfico y esperamos dos segundos
   ChartRedraw();
   Sleep(2000);
//--- movemos a la esquina inferior derecha
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_LOWER))
      return;
//--- cambiamos el texto de la etiqueta
   StringConcatenate(text,"La esquina inferior derecha: ",x,",",y);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redibujamos el gráfico y esperamos dos segundos
   ChartRedraw();
   Sleep(2000);
//--- movemos a la esquina superior derecha
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_UPPER))
      return;
//--- cambiamos el texto de la etiqueta
   StringConcatenate(text,"La esquina superior derecha: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redibujamos el gráfico y esperamos dos segundos
   ChartRedraw();
   Sleep(2000);
//--- movemos a la esquina superior izquierda
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_UPPER))
      return;
//--- cambiamos el texto de la etiqueta
   StringConcatenate(text,"La esquina superior izquierda: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redibujamos el gráfico y esperamos dos segundos
   ChartRedraw();
   Sleep(2000);
//--- eliminamos la etiqueta
   LabelDelete(0,InpName);
//--- redibujamos el gráfico y esperamos medio segundo
   ChartRedraw();
   Sleep(500);
//---
  }
//+------------------------------------------------------------------+
//| La función mueve el objeto y cambia su texto                     |
//+------------------------------------------------------------------+
bool MoveAndTextChange(const int x,const int y,string text)
  {
//--- movemos la etiqueta
   if(!LabelMove(0,InpName,x,y))
      return(false);
//--- cambiamos el texto de la etiqueta
   StringConcatenate(text,text,x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return(false);
//--- comprobamos si el trabajo del script ha sido finalizado forzosamente
   if(IsStopped())
      return(false);
//--- redibujamos el gráfico
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
// retardo de 0,01 segundo
   Sleep(10);
//--- salida de la función
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BUTTON

Objeto "Botón".



Nota

Las coordenadas del punto de anclaje se establecen en píxeles. Desde la enumeración ENUM_BASE_CORNER se puede elegir la esquina de enlace del botón.

Ejemplo

El siguiente script crea y desplaza el objeto "Botón" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea el botón en el gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string           InpName="Button";           // Nombre del botón
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Esquina del gráfico para el en
input string           InpFont="Arial";            // Fuente
input int              InpFontSize=14;             // Tamaño de la fuente
input color            InpColor=clrBlack;          // Color del texto
input color            InpBackColor=C'236,233,216'; // Color del fondo
input color            InpBorderColor=clrNONE;     // Color del borde
input bool             InpState=false;             // Pulsado/No pulsado
input bool             InpBack=false;              // Objeto al fondo
input bool             InpSelection=false;         // Seleccionar para mover
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpHidden=true;              // Ocultar en la lista de objetos
input long              InpZOrder=0;                 // Prioridad para el clic del rat
//+------------------------------------------------------------------+
//| Crea el botón                                                    |
//+------------------------------------------------------------------+
bool ButtonCreate(const long              chart_ID=0,               // ID del gráfico
                  const string            name="Button",            // nombre del bot
                  const int               sub_window=0,             // número de subve
                  const int               x=0,                      // coordenada por
                  const int               y=0,                      // coordenada por
                  const int               width=50,                 // ancho del botó
                  const int               height=18,                // alto del botón
                  const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // esquina del gra
                  const string            text="Button",            // texto
                  const string            font="Arial",             // fuente
                  const int               font_size=10,             // tamaño de la fu
                  const color             clr=clrBlack,             // color del texto
                  const color             back_clr=C'236,233,216',  // color del fondo
                  const color             border_clr=clrNONE,       // color del borde
                  const bool              state=false,              // pulsado/no puls
                  const bool              back=false,               // al fondo
                  const bool              selection=false,          // seleccionar par
                  const bool              hidden=true,              // ocultar en la l
                  const long              z_order=0)                //prioridad para e
  {
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el botón
   if(!ObjectCreate(chart_ID,name,OBJ_BUTTON,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el botón! Código del error = ",GetLastError());
      return(false);
     }
//--- establecemos las coordenadas del botón
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- establecemos el tamaño del botón
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- establecemos la esquina del gráfico respecto a la cual van a determinarse las co
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- ponemos el texto
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- establecemos la fuente del texto
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- establecemos el tamaño del texto
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- establecemos el color del texto
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el color del fondo
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- establecemos el color del borde
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- set button state
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- activar (true) o desactivar (false) el modo de desplazamiento del botón con rató
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el botón                                                   |
//+------------------------------------------------------------------+
bool ButtonMove(const long   chart_ID=0,    // ID del gráfico
                const string name="Button", // nombre del botón
                const int    x=0,           // coordenada por el eje X
                const int    y=0)           // coordenada por el eje Y
  {
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el botón
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada X del botón! Código del error = ",GetLast
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada Y del botón! Código del error = ",GetLast
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el tamaño del botón                                       |
//+------------------------------------------------------------------+
bool ButtonChangeSize(const long   chart_ID=0,    // ID del gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
                      const string name="Button", // nombre del botón
                      const int    width=50,      // ancho del botón
                      const int    height=18)     // alto del botón
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos las dimensiones del botón
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el ancho del botón! Código del error = ",GetLastError
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el alto del botón! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| //| Cambia la esquina del gráfico para el enlace del botón
//+------------------------------------------------------------------+
bool ButtonChangeCorner(const long             chart_ID=0,               // ID del grá
                        const string           name="Button",            // nombre del
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // esquina de
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos la esquina de anclaje
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la esquina de anclaje! Código del error = ",GetLastEr
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el texto del botón                                        |
//+------------------------------------------------------------------+
bool ButtonTextChange(const long   chart_ID=0,    // ID del gráfico
                      const string name="Button", // nombre del botón
                      const string text="Text")   // texto
  {
//--- anulamos el valor del error
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- cambiamos el texto del objeto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el texto! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el botón                                                 |
//+------------------------------------------------------------------+
bool ButtonDelete(const long   chart_ID=0,    // ID del gráfico
                  const string name="Button") // nombre del botón
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el botón
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el botón! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tamaño de la ventana del gráfico
   long x_distance;
   long y_distance;
//--- definimos las dimensiones de la ventana
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("¡Fallo al obtener el ancho del gráfico! Código del error = ",GetLastError
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("¡Fallo al obtener el alto del gráfico! Código del error = ",GetLastError
      return;
     }
//--- definimos el paso para el cambio del tamaño del botón
```

© 2000-2025, MetaQuotes Ltd.

```
   int x_step=(int)x_distance/32;
   int y_step=(int)y_distance/32;
//--- establecemos las coordenadas del botón y su tamaño
   int x=(int)x_distance/32;
   int y=(int)y_distance/32;
   int x_size=(int)x_distance*15/16;
   int y_size=(int)y_distance*15/16;
//--- creamos el botón
   if(!ButtonCreate(0,InpName,0,x,y,x_size,y_size,InpCorner,"Press",InpFont,InpFontSiz
      InpColor,InpBackColor,InpBorderColor,InpState,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
//--- redibujamos el gráfico
   ChartRedraw();
//--- en el ciclo reducimos el botón
   int i=0;
   while(i<13)
     {
      //--- retardo de medio segundo
      Sleep(500);
      //--- ponemos el botón en el estado "pulsado"
      ObjectSetInteger(0,InpName,OBJPROP_STATE,true);
      //--- redibujamos el gráfico y esperamos 0,2 segundo
      ChartRedraw();
      Sleep(200);
      //--- redefinimos las coordenadas y el tamaño del botón
      x+=x_step;
      y+=y_step;
      x_size-=x_step*2;
      y_size-=y_step*2;
      //--- reducimos el botón
      ButtonMove(0,InpName,x,y);
      ButtonChangeSize(0,InpName,x_size,y_size);
      //--- volveremos el botón en el estado de "no pulsado"
      ObjectSetInteger(0,InpName,OBJPROP_STATE,false);
      //--- redibujamos el gráfico
      ChartRedraw();
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- aumentamos el contador del ciclo
      i++;
     }
//--- retardo de medio segundo
   Sleep(500);
//--- eliminamos el botón
   ButtonDelete(0,InpName);
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- esperamos 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_CHART

Objeto "Gráfico".



Nota

El objeto de tipo "OBJ_CHART" no tiene soporte (no se representa) en las pruebas visuales.

Las coordenadas del punto de anclaje se establecen en píxeles. Se puede elegir la esquina de enlace desde la enumeración ENUM_BASE_CORNER.

Para el objeto "Gráfico" se puede elegir el símbolo, período y la escala, así como activar/desactivar el modo de visualización de la escala del precio y de la fecha.

Ejemplo

El siguiente script crea y desplaza el objeto "Gráfico" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea el objeto \"Gráfico\"."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string           InpName="Chart";              // Nombre del objeto
input string           InpSymbol="EURUSD";           // Símbolo
input ENUM_TIMEFRAMES  InpPeriod=PERIOD_H1;          // Período
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Esquina para el enlace
input int              InpScale=2;                   // Escala
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpDateScale=true;           // Visualización de la escala del
input bool              InpPriceScale=true;          // Visualización de la escala del
input color             InpColor=clrRed;             // Color del borde durante la sele
input ENUM_LINE_STYLE   InpStyle=STYLE_DASHDOTDOT;   // Estilo de la línea durante la s
input int               InpPointWidth=1;             // Tamaño del punto para los movin
input bool              InpBack=false;               // Objeto al fondo
input bool              InpSelection=true;           // Seleccionar para mover
input bool              InpHidden=true;              // Ocultar en la lista de objetos
input long              InpZOrder=0;                 // Prioridad para el clic del rató
//+------------------------------------------------------------------+
//| Crea el objeto "Gráfico"                                         |
//+------------------------------------------------------------------+
bool ObjectChartCreate(const long              chart_ID=0,               // ID del grá
                       const string            name="Chart",             // nombre del
                       const int               sub_window=0,             // número de
                       const string            symbol="EURUSD",          // símbolo
                       const ENUM_TIMEFRAMES   period=PERIOD_H1,         // período
                       const int               x=0,                      // coordenada
                       const int               y=0,                      // coordenada
                       const int               width=300,                // ancho
                       const int               height=200,               // alto
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // esquina pa
                       const int               scale=2,                  // escala
                       const bool              date_scale=true,          // visualizac
                       const bool              price_scale=true,         // isualizaci
                       const color             clr=clrRed,               // color del
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // estilo de
                       const int               point_width=1,            // tamaño del
                       const bool              back=false,               // al fondo
                       const bool              selection=false,          // selecciona
                       const bool              hidden=true,              // ocultar en
                       const long              z_order=0)                //prioridad p
  {
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el objeto "Gráfico"
   if(!ObjectCreate(chart_ID,name,OBJ_CHART,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el objeto \"Gráfico\"! Código del error = ",GetLastErro
      return(false);
     }
//--- establecemos las coordenadas del objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- establecemos el tamaño del objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- establecemos la esquina del gráfico respecto a la cual van a determinarse las co
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- establecemos el símbolo
   ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol);
//--- ponemos el período
   ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period);
//--- establecemos la escala
   ObjectSetInteger(chart_ID,name,OBJPROP_CHART_SCALE,scale);
//--- mostramos (true) u ocultamos (false) la escala del tiempo
   ObjectSetInteger(chart_ID,name,OBJPROP_DATE_SCALE,date_scale);
//--- mostramos (true) u ocultamos (false) la escala del precio
   ObjectSetInteger(chart_ID,name,OBJPROP_PRICE_SCALE,price_scale);
//--- establecemos el color del contorno durante el modo de selección del objeto activ
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo del contorno durante el modo de selección del objeto acti
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño del punto de enlace con el que se puede mover el objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la etiqueta cor
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Establece el símbolo y el período del objeto "Gráfico"           |
//+------------------------------------------------------------------+
bool ObjectChartSetSymbolAndPeriod(const long            chart_ID=0,        // ID del g
                                   const string          name="Chart",      // nombre d
                                   const string          symbol="EURUSD",   // símbolo
                                   const ENUM_TIMEFRAMES period=PERIOD_H1) // período
  {
//--- anulamos el valor del error
   ResetLastError();
//--- establecemos el símbolo y el período del objeto "Gráfico"
   if(!ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al establecer el símbolo para el objeto \"Gráfico\"! Código del
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": ¡Fallo al establecer el período para el objeto \"Gráfico\"! Código del
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el objeto "Gráfico"                                        |
//+------------------------------------------------------------------+
bool ObjectChartMove(const long   chart_ID=0,   // ID del gráfico (no objeto)
                     const string name="Chart", // nombre del objeto
                     const int    x=0,          // coordenada por el eje X
                     const int    y=0)          // coordenada por el eje Y
  {
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada X del objeto \"Gráfico\"! Código del erro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada Y del objeto \"Gráfico\"! Código del erro
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el tamaño del objeto "Gráfico"                            |
//+------------------------------------------------------------------+
bool ObjectChartChangeSize(const long   chart_ID=0,   // ID del gráfico (no objeto)
                           const string name="Chart", // nombre del objeto
                           const int    width=300,    // ancho
                           const int    height=200)   // alto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos las dimensiones del objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el ancho del objeto \"Gráfico\"! Código del error = "
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el alto del objeto \"Gráfico\"! Código del error = ",
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Devuelve ID del objeto "Gráfico"                                 |
//+------------------------------------------------------------------+
long ObjectChartGetID(const long   chart_ID=0,   // ID del gráfico (no objeto)
                      const string name="Chart") // nombre del objeto
  {
//--- preparamos la variable para recibir ID del objeto "Gráfico"
   long id=-1;
//--- anulamos el valor del error
   ResetLastError();
//--- recibimos ID
   if(!ObjectGetInteger(chart_ID,name,OBJPROP_CHART_ID,0,id))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al recibir ID del objeto \"Gráfico\"! Código del error = ",GetLa
     }
//--- devolución del resultado
   return(id);
  }
//+------------------------------------------------------------------+
//| Elimina el objeto "Gráfico"                                      |
//+------------------------------------------------------------------+
bool ObjectChartDelete(const long   chart_ID=0,   // ID del gráfico (no objeto)
                       const string name="Chart") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el botón
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el objeto \"Gráfico\"! Código del error = ",GetLastE
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- obtenemos el número de símbolos en la ventana "Observación del mercado"
   int  symbols=SymbolsTotal(true);
//--- comprobamos si hay el símbolo con el nombre especificado en la lista de los símb
   bool exist=false;
   for(int i=0;i<symbols;i++)
      if(InpSymbol==SymbolName(i,true))
        {
         exist=true;
         break;
        }
   if(!exist)
     {
      Print("¡Error. Este símbolo ",InpSymbol," no está en la ventana \"Observación de
      return;
     }
//--- comprobando si los parámetros de entrada son correctos
   if(InpScale<0 || InpScale>5)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }

//--- tamaño de la ventana del gráfico
   long x_distance;
   long y_distance;
//--- definimos las dimensiones de la ventana
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("¡Fallo al obtener el ancho del gráfico! Código del error = ",GetLastErro
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("¡Fallo al obtener el alto del gráfico! Código del error = ",GetLastError
      return;
     }
//--- establecemos las coordenadas del objeto "Gráfico" y su tamaño
   int x=(int)x_distance/16;
   int y=(int)y_distance/16;
   int x_size=(int)x_distance*7/16;
   int y_size=(int)y_distance*7/16;
//--- creamos el objeto "Gráfico"
   if(!ObjectChartCreate(0,InpName,0,InpSymbol,InpPeriod,x,y,x_size,y_size,InpCorner,I
      InpPriceScale,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
```

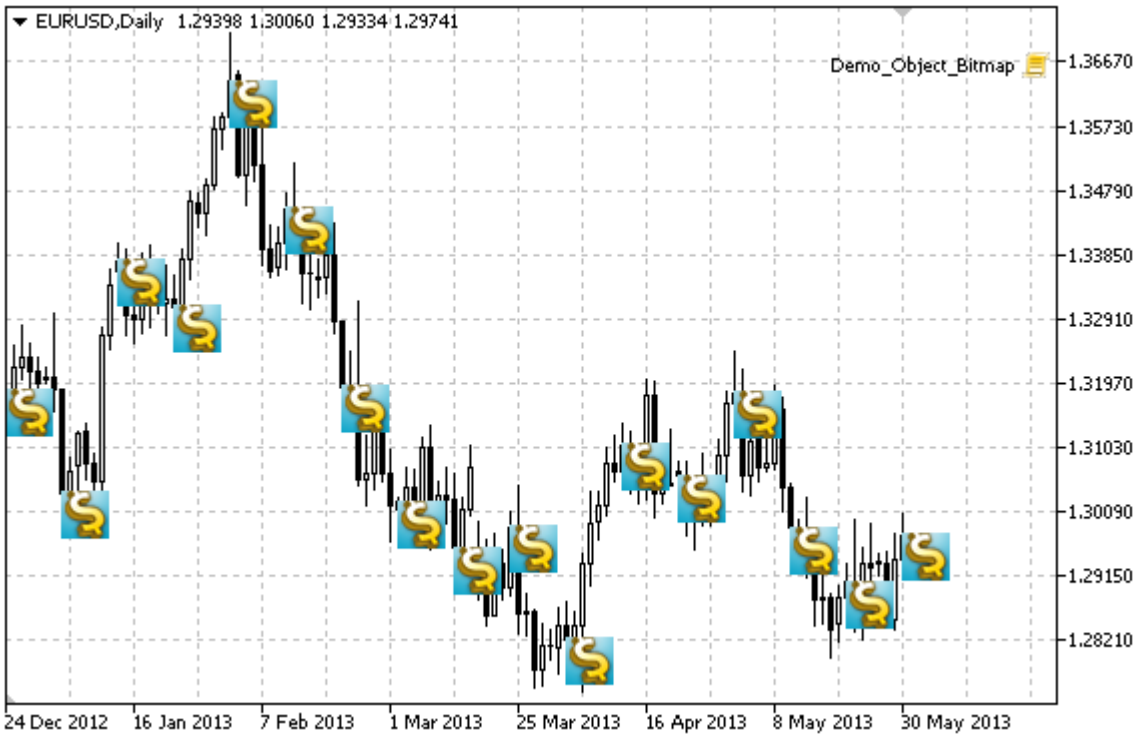
© 2000-2025, MetaQuotes Ltd.

```
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- extendemos el objeto "Gráfico"
   int steps=(int)MathMin(x_distance*7/16,y_distance*7/16);
   for(int i=0;i<steps;i++)
     {
      //--- cambiamos el tamaño
      x_size+=1;
      y_size+=1;
      if(!ObjectChartChangeSize(0,InpName,x_size,y_size))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico y esperamos 0,01 segundo
      ChartRedraw();
      Sleep(10);
     }
//--- retardo de medio segundo
   Sleep(500);
//--- cambiamos el período del gráfico
   if(!ObjectChartSetSymbolAndPeriod(0,InpName,InpSymbol,PERIOD_M1))
      return;
   ChartRedraw();
//--- retardo de tres segundos
   Sleep(3000);
//--- eliminamos el objeto
   ObjectChartDelete(0,InpName);
   ChartRedraw();
//--- esperamos 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BITMAP

Objeto "Dibujo".



Nota

Para el objeto "Dibujo" se puede elegir la zona de visibilidad de la imagen.

Ejemplo

El siguiente script crea varias imágenes en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea el dibujo en la ventana del gráfico."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpFile="\\Images\\dollar.bmp"; // Nombre del archivo con la ima
input int             InpWidth=24;                    // coordenada X de la zona de vi
input int             InpHeight=24;                   // coordenada Y de la zona de vi
input int             InpXOffset=4;                   // Desplazamiento de la zona de
input int             InpYOffset=4;                   // Desplazamiento de la zona de
input color           InpColor=clrRed;                // Color del borde durante la se
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;           // Estilo de la línea durante la
input int             InpPointWidth=1;                // Tamaño del punto para los mov
input bool            InpBack=false;                  // Objeto al fondo
input bool            InpSelection=false;             // Seleccionar para mover
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpHidden=true;                  // Ocultar en la lista de objeto
input long            InpZOrder=0;                     // Prioridad para el clic del ra
//+------------------------------------------------------------------+
//| Crea el dibujo en la ventana del gráfico                         |
//+------------------------------------------------------------------+
bool BitmapCreate(const long            chart_ID=0,        // ID del gráfico
                  const string          name="Bitmap",     // nombre del dibujo
                  const int             sub_window=0,      // número de subventana
                  datetime              time=0,            // hora del punto de ancla
                  double                price=0,           // precio del punto de anc
                  const string          file="",           // nombre del archivo con
                  const int             width=10,          // coordenada X de la zona
                  const int             height=10,         // coordenada Y de la zona
                  const int             x_offset=0,        // desplazamiento de la zo
                  const int             y_offset=0,        // desplazamiento de la zo
                  const color           clr=clrRed,        // color del borde durante
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de la línea dura
                  const int             point_width=1,     // tamaño del punto de des
                  const bool            back=false,        // al fondo
                  const bool            selection=false,   // seleccionar para mover
                  const bool            hidden=true,       // ocultar en la lista de
                  const long            z_order=0)         // prioridad para el clic
  {
//--- establecemos las coordenadas del punto de anclaje si todavía no han sido estable
   ChangeBitmapEmptyPoint(time,price);
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el dibujo
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el dibujo en la ventana del gráfico! Código del error =
      return(false);
     }
//--- establecemos la ruta hacia el archivo con la imagen
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cargar la imagen! Código del error = ",GetLastError());
      return(false);
     }
//--- establecemos la zona de visibilidad de la imagen; si los valores del ancho o del
//--- superan los valores del ancho o del alto (respectivamente) de la imagen original
//--- no se dibuja; si los valores del ancho o del alto son inferiores a las dimensiones
//--- se dibuja la parte que corresponde a estas dimensiones
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- establecemos la parte de la imagen que debe mostrarse en la zona de visibilidad
//--- por defecto se trata de la zona superior izquierda de la imagen; los valores per
```

© 2000-2025, MetaQuotes Ltd.

```
//--- hacer el desplazamiento de esta esquina y mostrar otra parte de la imagen
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- establecemos el color del contorno durante el modo de selección del objeto activ
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo del contorno durante el modo de selección del objeto acti
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño del punto de enlace con el que se puede mover el objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la etiqueta cor
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Pone nueva imagen para el dibujo                                 |
//+------------------------------------------------------------------+
bool BitmapSetImage(const long   chart_ID=0,    // ID del gráfico
                    const string name="Bitmap", // nombre de la imagen
                    const string file="")       // ruta hacia el archivo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- establecemos la ruta hacia el archivo con la imagen
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cargar la imagen! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el dibujo dentro de la ventana del gráfico                 |
//+------------------------------------------------------------------+
bool BitmapMove(const long   chart_ID=0,    // ID del gráfico
                const string name="Bitmap", // nombre del dibujo
                datetime     time=0,        // hora del punto de anclaje
                double       price=0)       // precio del punto de anclaje
  {
//--- si las coordenadas del punto de anclaje no han sido establecidas, lo movemos a l
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el punto de anclaje
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el punto de anclaje! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el tamaño de la zona de visibilidad (tamaño del dibujo)   |
//+------------------------------------------------------------------+
bool BitmapChangeSize(const long   chart_ID=0,     // ID del gráfico
                      const string name="Bitmap", // nombre del dibujo
                      const int    width=0,       // ancho del dibujo
                      const int    height=0)      // alto del dibujo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos las dimensiones del dibujo
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el ancho del dibujo! Código del error = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el alto del dibujo! Código del error = ",GetLastErro
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia la coordenada de la esquina superior izquierda de la zona de visibilidad
//+------------------------------------------------------------------+
bool BitmapMoveVisibleArea(const long   chart_ID=0,     // ID del gráfico
                           const string name="Bitmap", // nombre del dibujo
                           const int    x_offset=0,    // coordenada X de la zona de v
                           const int    y_offset=0)    // coordenada Y de la zona de v
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos las coordenadas de la zona de visibilidad del dibujo
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la coordenada X de la zona de visibilidad! Código del
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la coordenada Y de la zona de visibilidad! Código del
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el dibujo                                                |
//+------------------------------------------------------------------+
bool BitmapDelete(const long   chart_ID=0,    // ID del gráfico
                  const string name="Bitmap") // nombre del dibujo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el dibujo! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprueba los valores de los puntos de anclaje y para            |
//| los valores vacíos establece los valores por defecto             |
//+------------------------------------------------------------------+
void ChangeBitmapEmptyPoint(datetime &time,double &price)
  {
//--- si la hora del punto no ha sido establecida, se colocará en la barra actual
   if(!time)
      time=TimeCurrent();
//--- si el precio del punto no ha sido establecido, tendrá el valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[];  // array para guardar las fechas de las barras visibles
   double   close[]; // array para guardar los precios Close
//--- nombre del archivo con la imagen
   string   file="\\Images\\dollar.bmp";
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- asignación de la memoria
   ArrayResize(date,bars);
   ArrayResize(close,bars);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- llenamos el array de precios Close
   if(CopyClose(Symbol(),Period(),0,bars,close)==-1)
     {
      Print("¡Fallo al copiar los valores de los precios Close! Código del error = ",G
      return;
     }
//--- definimos con qué frecuencia hay que mostrar las imágenes
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- definimos el paso
   int step=1;
   switch(scale)
     {
      case 0:
         step=27;
         break;
      case 1:
         step=14;
         break;
      case 2:
         step=7;
         break;
      case 3:
         step=4;
         break;
      case 4:
         step=2;
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- creamos los dibujos para los valores High y Low de las barras (con intervalos)
   for(int i=0;i<bars;i+=step)
     {
      //--- creamos los dibujos
      if(!BitmapCreate(0,"Bitmap_"+(string)i,0,date[i],close[i],InpFile,InpWidth,InpHe
         InpYOffset,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
        {
         return;
        }
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de medio segundo
   Sleep(500);
//--- eliminamos los signos "Sell"
   for(int i=0;i<bars;i+=step)
     {
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BITMAP_LABEL

Objeto "Etiqueta gráfica".



Nota

Desde la enumeración ENUM_ANCHOR_POINT se puede elegir la posición del punto de anclaje respecto a la etiqueta. Las coordenadas del punto de anclaje se establecen en píxeles.

Además, desde la enumeración ENUM_BASE_CORNER se puede elegir la esquina de enlace de la etiqueta gráfica.

Para el objeto "Etiqueta gráfica" se puede elegir la zona de visibilidad de la imagen.

Ejemplo

El siguiente script crea varias imágenes en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea el objeto \"Etiqueta gráfica\"."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="BmpLabel";              // Nombre de la etiqueta
input string            InpFileOn="\\Images\\dollar.bmp"; // Nombre del archivo para e
input string            InpFileOff="\\Images\\euro.bmp";  // Nombre del archivo para e
input bool              InpState=false;                   // Etiqueta pulsada/no pulsa
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER;      // Esquina del gráfico para
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER;          // Modo del enlace
```

© 2000-2025, MetaQuotes Ltd.

```
input color             InpColor=clrRed;                  // Color del borde durante l
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;             // Estilo de la línea durant
input int               InpPointWidth=1;                  // Tamaño del punto para los
input bool              InpBack=false;                    // Objeto al fondo
input bool              InpSelection=false;               // Seleccionar para mover
input bool              InpHidden=true;                   // Ocultar en la lista de ob
input long              InpZOrder=0;                      // Prioridad para el clic de
//+------------------------------------------------------------------+
//| Crea el objeto "Etiqueta gráfica"                                |
//+------------------------------------------------------------------+
bool BitmapLabelCreate(const long              chart_ID=0,               // ID del grá
                       const string            name="BmpLabel",          // nombre de
                       const int               sub_window=0,             // número de
                       const int               x=0,                      // coordenada
                       const int               y=0,                      // coordenada
                       const string            file_on="",               // imagen en
                       const string            file_off="",              // imagen en
                       const int               width=0,                  // coordenada
                       const int               height=0,                 // coordenada
                       const int               x_offset=10,              // desplazami
                       const int               y_offset=10,              // desplazami
                       const bool              state=false,              // pulsada/no
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // esquina de
                       const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // modo de en
                       const color             clr=clrRed,               // color del
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // estilo de
                       const int               point_width=1,            // tamaño del
                       const bool              back=false,               // al fondo
                       const bool              selection=false,          // selecciona
                       const bool              hidden=true,              // ocultar en
                       const long              z_order=0)                //prioridad p
  {
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la etiqueta gráfica
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el objeto \"Etiqueta gráfica\"! Código del error = ",Ge
      return(false);
     }
//--- ponemos las imágenes para los modos On y Off
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,0,file_on))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cargar la imagen para el modo On! Código del error = ",GetLas
      return(false);
     }
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,1,file_off))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": ¡Fallo al cargar la imagen para el modo Off! Código del error = ",GetLa
       return(false);
      }
//--- establecemos las coordenadas de la etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- establecemos la zona de visibilidad de la imagen; si los valores del ancho o del
//--- superan los valores del ancho o del alto (respectivamente) de la imagen original
//--- no se dibuja; si los valores del ancho o del alto son inferiores a las dimensior
//--- se dibuja la parte que corresponde a estas dimensiones
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- establecemos la parte de la imagen que debe mostrarse en la zona de visibilidad
//--- por defecto se trata de la zona superior izquierda de la imagen; los valores per
//--- hacer el desplazamiento de esta esquina y mostrar otra parte de la imagen
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- definimos en qué estado se encuentra la etiqueta (pulsada o no pulsada)
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- establecemos la esquina del gráfico respecto a la cual van a determinarse las co
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- establecemos el modo de anclaje
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- establecemos el color del contorno durante el modo de selección del objeto activ
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo del contorno durante el modo de selección del objeto acti
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el tamaño del punto de enlace con el que se puede mover el objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la etiqueta con
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Pone nueva imagen para el objeto "Etiqueta gráfica"              |
//+------------------------------------------------------------------+
bool BitmapLabelSetImage(const long   chart_ID=0,      // ID del gráfico
                         const string name="BmpLabel", // nombre de la etiqueta
                         const int    on_off=0,        // modificador (On u Off)
```

© 2000-2025, MetaQuotes Ltd.

```
                          const string file="")         // ruta hacia el archivo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- establecemos la ruta hacia el archivo con la imagen
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,on_off,file))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cargar la imagen! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el objeto "Etiqueta gráfica"                               |
//+------------------------------------------------------------------+
bool BitmapLabelMove(const long   chart_ID=0,      // ID del gráfico
                     const string name="BmpLabel", // nombre de la etiqueta
                     const int    x=0,             // coordenada por el eje X
                     const int    y=0)             // coordenada por el eje Y
  {
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada X del objeto! Código del error = ",GetLas
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada X del objeto! Código del error = ",GetLas
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el tamaño de la zona de visibilidad (tamaño del objeto)   |
//+------------------------------------------------------------------+
bool BitmapLabelChangeSize(const long   chart_ID=0,      // ID del gráfico
                           const string name="BmpLabel", // nombre de la etiqueta
                           const int    width=0,         // ancho de la etiqueta
                           const int    height=0)        // alto de la etiqueta
  {
//--- anulamos el valor del error
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- cambiamos las dimensiones del objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el ancho del objeto! Código del error = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el alto del objeto! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia la coordenada de la esquina superior izquierda de la zona de visibilidad
//+------------------------------------------------------------------+
bool BitmapLabelMoveVisibleArea(const long   chart_ID=0,      // ID del gráfico
                                const string name="BmpLabel", // nombre de la etiqueta
                                const int    x_offset=0,      // coordenada X de la zo
                                const int    y_offset=0)      // coordenada Y de la zo
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos las coordenadas de la zona de visibilidad del objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la coordenada X de la zona de visibilidad! Código del
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar la coordenada Y de la zona de visibilidad! Código del
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el objeto "Etiqueta gráfica"                             |
//+------------------------------------------------------------------+
bool BitmapLabelDelete(const long   chart_ID=0,      // ID del gráfico
                       const string name="BmpLabel") // nombre de la etiqueta
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el objeto \"Etiqueta gráfica\"! Código del error = "
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tamaño de la ventana del gráfico
   long x_distance;
   long y_distance;
//--- definimos las dimensiones de la ventana
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("¡Fallo al obtener el ancho del gráfico! Código del error = ",GetLastErro
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("¡Fallo al obtener el alto del gráfico! Código del error = ",GetLastError
      return;
     }
//--- definimos las coordenadas de la etiqueta gráfica
   int x=(int)x_distance/2;
   int y=(int)y_distance/2;
//--- establecemos las dimensiones de la etiqueta y las coordenadas de la zona de visi
   int width=32;
   int height=32;
   int x_offset=0;
   int y_offset=0;
//--- colocamos la etiqueta gráfica en el centro de la ventana
   if(!BitmapLabelCreate(0,InpName,0,x,y,InpFileOn,InpFileOff,width,height,x_offset,y_
      InpCorner,InpAnchor,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHido
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- cambiamos el tamaño de la zona de visibilidad de la etiqueta en el ciclo
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<6;i++)
     {
      //--- cambiamos el tamaño de la zona de visibilidad
      width--;
      height--;
      if(!BitmapLabelChangeSize(0,InpName,width,height))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,3 segundo
      Sleep(300);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- cambiamos las coordenadas de la zona de visibilidad de la etiqueta en el ciclo
   for(int i=0;i<2;i++)
     {
      //--- cambiamos las coordenadas de la zona de visibilidad
      x_offset++;
      y_offset++;
      if(!BitmapLabelMoveVisibleArea(0,InpName,x_offset,y_offset))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,3 segundo
      Sleep(300);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos la etiqueta
   BitmapLabelDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EDIT

Objeto "Campo de edición".



Nota

Las coordenadas del punto de anclaje se establecen en píxeles. Desde la enumeración ENUM_BASE_CORNER se puede elegir la esquina de enlace del campo de edición.

Además, se puede elegir uno de los tipos de alineación del texto dentro del "Campo de edición" usando la enumeración ENUM_ALIGN_MODE.

Ejemplo

El siguiente script crea y desplaza la "Campo de edición" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea el objeto \"Campo de edición\"."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string            InpName="Edit";              // Nombre del objeto
input string            InpText="Text";              // Texto del objeto
input string            InpFont="Arial";             // Fuente
input int               InpFontSize=14;              // Tamaño de la fuente
input ENUM_ALIGN_MODE   InpAlign=ALIGN_CENTER;       // Modo de alineación del texto
input bool              InpReadOnly=false;           // Posibilidad de edición
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER; // Esquina del gráfico para el enl
input color             InpColor=clrBlack;           // Color del texto
```

© 2000-2025, MetaQuotes Ltd.

```
input color             InpBackColor=clrWhite;       // Color del fondo
input color             InpBorderColor=clrBlack;     // Color del borde
input bool              InpBack=false;               // Objeto al fondo
input bool              InpSelection=false;          // Seleccionar para mover
input bool              InpHidden=true;              // Ocultar en la lista de objetos
input long              InpZOrder=0;                 // Prioridad para el clic del rat
//+------------------------------------------------------------------+
//| Crea el objeto "Campo de edición"                                |
//+------------------------------------------------------------------+
bool EditCreate(const long             chart_ID=0,               // ID del gráfico
                const string           name="Edit",              // nombre del objeto
                const int              sub_window=0,             // número de subventa
                const int              x=0,                      // coordenada por el
                const int              y=0,                      // coordenada por el
                const int              width=50,                 // ancho
                const int              height=18,                // alto
                const string           text="Text",              // texto
                const string           font="Arial",             // fuente
                const int              font_size=10,             // tamaño de la fuent
                const ENUM_ALIGN_MODE  align=ALIGN_CENTER,       // modo de alineació
                const bool             read_only=false,          // posibilidad de ed
                const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // esquina del gráfi
                const color            clr=clrBlack,             // color del texto
                const color            back_clr=clrWhite,        // color del fondo
                const color            border_clr=clrNONE,       // color del borde
                const bool             back=false,               // al fondo
                const bool             selection=false,          // seleccionar para
                const bool             hidden=true,              // ocultar en la lis
                const long             z_order=0)                // prioridad para el
  {
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el campo de edición
   if(!ObjectCreate(chart_ID,name,OBJ_EDIT,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el objeto \"Campo de edición\"! Código del error = ",Ge
      return(false);
     }
//--- establecemos las coordenadas del objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- establecemos el tamaño del objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- ponemos el texto
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- establecemos la fuente del texto
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- establecemos el tamaño del texto
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- establecemos el modo de alineación del texto dentro del objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_ALIGN,align);
//--- ponemos (true) o cancelamos (false) el modo sólo para lectura
   ObjectSetInteger(chart_ID,name,OBJPROP_READONLY,read_only);
//--- establecemos la esquina del gráfico respecto a la cual van a determinarse las co
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- establecemos el color del texto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el color del fondo
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- establecemos el color del borde
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la etiqueta cor
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el objeto "Campo de edición"                               |
//+------------------------------------------------------------------+
bool EditMove(const long   chart_ID=0,  // ID del gráfico
              const string name="Edit", // nombre del objeto
              const int    x=0,         // coordenada por el eje X
              const int    y=0)         // coordenada por el eje Y
  {
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada X del objeto! Código del error = ",GetLas
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada X del objeto! Código del error = ",GetLas
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el tamaño del objeto "Campo de edición"                   |
//+------------------------------------------------------------------+
bool EditChangeSize(const long   chart_ID=0,  // ID del gráfico
                    const string name="Edit", // nombre del objeto
                    const int    width=0,     // ancho
                    const int    height=0)    // alto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos las dimensiones del objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el ancho del objeto! Código del error = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el alto del objeto! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el texto del objeto "Campo de edición"                    |
//+------------------------------------------------------------------+
bool EditTextChange(const long   chart_ID=0,  // ID del gráfico
                    const string name="Edit", // nombre del objeto
                    const string text="Text") // texto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el texto del objeto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el texto! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Devuelve el texto del objeto "Campo de edición"                          |
//+------------------------------------------------------------------+
bool EditTextGet(string      &text,        // texto
                 const long   chart_ID=0,  // ID del gráfico
                 const string name="Edit") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- recibimos el texto del objeto
   if(!ObjectGetString(chart_ID,name,OBJPROP_TEXT,0,text))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al recibir el texto! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el objeto "Campo de edición"                             |
//+------------------------------------------------------------------+
bool EditDelete(const long   chart_ID=0,  // ID del gráfico
                const string name="Edit") // nombre del objeto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el objeto \"Campo de edición\"! Código del error = "
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tamaño de la ventana del gráfico
   long x_distance;
   long y_distance;
//--- definimos las dimensiones de la ventana
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("¡Fallo al obtener el ancho del gráfico! Código del error = ",GetLastErro
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
      {
       Print("¡Fallo al obtener el alto del gráfico! Código del error = ",GetLastError
       return;
      }
//--- definimos el paso para el cambio del tamaño del campo de edición
   int x_step=(int)x_distance/64;
//--- establecemos las coordenadas del campo de edición y su tamaño
   int x=(int)x_distance/8;
   int y=(int)y_distance/2;
   int x_size=(int)x_distance/8;
   int y_size=InpFontSize*2;
//--- recordamos el texto en la variable local
   string text=InpText;
//--- creamos el campo de edición
   if(!EditCreate(0,InpName,0,x,y,x_size,y_size,InpText,InpFont,InpFontSize,InpAlign,I
       InpCorner,InpColor,InpBackColor,InpBorderColor,InpBack,InpSelection,InpHidden,In
      {
       return;
      }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- extendemos el campo de edición
   while(x_size-x<x_distance*5/8)
      {
       //--- aumentamos el ancho del campo de edición
       x_size+=x_step;
       if(!EditChangeSize(0,InpName,x_size,y_size))
          return;
       //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
       if(IsStopped())
          return;
       //--- redibujamos el gráfico y esperamos 0,05 segundo
       ChartRedraw();
       Sleep(50);
      }
//--- retardo de medio segundo
   Sleep(500);
//--- cambiamos el texto
   for(int i=0;i<20;i++)
      {
       //--- añadimos "+" al principio y al final
       text="+"+text+"+";
       if(!EditTextChange(0,InpName,text))
          return;
       //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
       if(IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
       //--- redibujamos el gráfico y esperamos 0,1 segundo
       ChartRedraw();
       Sleep(100);
      }
//--- retardo de medio segundo
   Sleep(500);
//--- eliminamos el campo de edición
   EditDelete(0,InpName);
   ChartRedraw();
//--- esperamos 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EVENT

Objeto "Evento".



Nota

Al apuntar el cursor sobre el evento, se muestra su texto.

Ejemplo

El siguiente script crea y desplaza el objeto "Evento" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script construye el objeto gráfico \"Evento\"."
#property description "La fecha del punto de anclaje se establece en por cientos del"
#property description "ancho de la ventana del gráfico de barras."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string          InpName="Event";     // Nombre del evento
input int             InpDate=25;          // Fecha del evento en %
input string          InpText="Text";      // Texto del evento
input color           InpColor=clrRed;     // Color del evento
input int             InpWidth=1;          // Tamaño del punto durante la selección
input bool            InpBack=false;       // Evento al fondo
input bool            InpSelection=false; // Seleccionar para mover
input bool            InpHidden=true;      // Ocultar en la lista de objetos
```

© 2000-2025, MetaQuotes Ltd.

```
input long            InpZOrder=0;        // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea el objeto "Evento" en el gráfico                            |
//+------------------------------------------------------------------+
bool EventCreate(const long            chart_ID=0,       // ID del gráfico
                 const string          name="Event",     // nombre del evento
                 const int             sub_window=0,     // número de subventana
                 const string          text="Text",      // texto del evento
                 datetime              time=0,           // hora
                 const color           clr=clrRed,       // color
                 const int             width=1,          // grosor del punto durante la
                 const bool            back=false,       // al fondo
                 const bool            selection=false, // seleccionar para mover
                 const bool            hidden=true,      // ocultar en la lista de obje
                 const long            z_order=0)        // prioridad para el clic del
  {
//--- si la hora no está definida, creamos el objeto en la última barra
   if(!time)
      time=TimeCurrent();
//--- anulamos el valor del error
   ResetLastError();
//--- creamos el objeto "Evento"
   if(!ObjectCreate(chart_ID,name,OBJ_EVENT,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear el objeto \"Evento\"! Código del error = ",GetLastError
      return(false);
     }
//--- ponemos el texto del evento
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- establecemos el color
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el grosor del punto de anclaje si el objeto está seleccionado
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento del evento con rat
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el texto del objeto "Evento"                              |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool EventTextChange(const long   chart_ID=0,   // ID del gráfico
                     const string name="Event", // nombre del evento
                     const string text="Text")  // texto
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el texto del objeto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el texto! Código del error = ",GetLastError());
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve el objeto "Evento"                                         |
//+------------------------------------------------------------------+
bool EventMove(const long   chart_ID=0,   // ID del gráfico
               const string name="Event", // nombre del evento
               datetime     time=0)       // hora
  {
//--- si la hora no está definida, movemos el evento a la última barra
   if(!time)
      time=TimeCurrent();
//--- anulamos el valor del error
   ResetLastError();
//--- movemos el objeto
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover el objeto \"Evento\"! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina el objeto "Evento"                                       |
//+------------------------------------------------------------------+
bool EventDelete(const long   chart_ID=0,   // ID del gráfico
                 const string name="Event") // nombre del evento
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos el objeto
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": ¡Fallo al eliminar el objeto \"Evento\"! Código del error = ",GetLastEr
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos si los parámetros de entrada son correctos
   if(InpDate<0 || InpDate>100)
     {
      Print("¡Error. Los parámetros de entrada no son correctos!");
      return;
     }
//--- número de barras visibles en la ventana del gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- array para guardar los valores de las fechas que van a utilizarse
//--- para establecer y cambiar las coordenadas del punto de anclaje de el objeto
   datetime date[];
//--- asignación de la memoria
   ArrayResize(date,bars);
//--- llenamos el array de datos
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("¡Fallo al copiar el valor de la hora! Código del error = ",GetLastError()
      return;
     }
//--- definimos los puntos para crear el objeto
   int d=InpDate*(bars-1)/100;
//--- creamos el objeto "Evento"
   if(!EventCreate(0,InpName,0,InpText,date[d],InpColor,InpWidth,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- ahora vamos a mover el objeto
//--- contador del ciclo
   int h_steps=bars/2;
//--- movemos el objeto
   for(int i=0;i<h_steps;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- cogemos el siguiente valor
      if(d<bars-1)
         d+=1;
      //--- movemos el punto
      if(!EventMove(0,InpName,date[d]))
         return;
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico
      ChartRedraw();
      // retardo de 0,05 segundo
      Sleep(50);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- eliminamos el canal desde el gráfico
   EventDelete(0,InpName);
   ChartRedraw();
//--- retardo de 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_RECTANGLE_LABEL

Objeto "Etiqueta rectangular".



Nota

Las coordenadas del punto de anclaje se establecen en píxeles. Desde la enumeración ENUM_BASE_CORNER se puede elegir la esquina de enlace de la etiqueta rectangular. Desde la enumeración ENUM_BORDER_TYPE se puede elegir el tipo de los bordes de la etiqueta rectangular.

Este objeto se utiliza para crear y diseñar la interfaz gráfica personalizada.

Ejemplo

El siguiente script crea y desplaza la "Etiqueta rectangular" en el gráfico. Para la creación y modificación de las propiedades del objeto gráfico han sido escritas unas funciones especiales que Usted puede utilizar "como son" en sus propias aplicaciones.

```
//--- descripción
#property description "El script crea el objeto gráfico \"Etiqueta rectangular\"."
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- los parámetros de entrada del script
input string           InpName="RectLabel";          // Nombre de la etiqueta
input color            InpBackColor=clrSkyBlue;      // Color del fondo
input ENUM_BORDER_TYPE InpBorder=BORDER_FLAT;        // Tipo del borde
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Esquina del gráfico para el en
input color            InpColor=clrDarkBlue;         // Color del contorno plano (Flat)
input ENUM_LINE_STYLE  InpStyle=STYLE_SOLID;         // Estilo del contorno plano (Flat
input int              InpLineWidth=3;               // Grosor del contorno plano (Flat
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpBack=false;              // Objeto al fondo
input bool              InpSelection=true;          // Seleccionar para mover
input bool              InpHidden=true;             // Ocultar en la lista de objetos
input long              InpZOrder=0;                // Prioridad para el clic del ratón
//+------------------------------------------------------------------+
//| Crea la etiqueta rectangular                                     |
//+------------------------------------------------------------------+
bool RectLabelCreate(const long             chart_ID=0,               // ID del gráfico
                     const string           name="RectLabel",         // nombre de la
                     const int              sub_window=0,             // número de sub
                     const int              x=0,                      // coordenada po
                     const int              y=0,                      // coordenada po
                     const int              width=50,                 // ancho
                     const int              height=18,                // alto
                     const color            back_clr=C'236,233,216',  // color del fon
                     const ENUM_BORDER_TYPE border=BORDER_SUNKEN,     // tipo del bord
                     const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // esquina del g
                     const color            clr=clrRed,               // color del cor
                     const ENUM_LINE_STYLE  style=STYLE_SOLID,        // estilo del co
                     const int              line_width=1,             // grosor del co
                     const bool             back=false,               // al fondo
                     const bool             selection=false,          // seleccionar p
                     const bool             hidden=true,              // ocultar en la
                     const long             z_order=0)                // prioridad par
  {
//--- anulamos el valor del error
   ResetLastError();
//--- creamos la etiqueta rectangular
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al crear la etiqueta rectangular! Código del error = ",GetLastEr
      return(false);
     }
//--- establecemos las coordenadas de la etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- establecemos las dimensiones de la etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- establecemos el color del fondo
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- establecemos el tipo del borde
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border);
//--- establecemos la esquina del gráfico respecto a la cual van a determinarse las co
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- establecemos el color del contorno plano (en modo Flat)
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- establecemos el estilo de las líneas del contorno plano
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- establecemos el grosor del contorno plano
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,line_width);
//--- mostramos en el primer plano (false) o al fondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activar (true) o desactivar (false) el modo de desplazamiento de la etiqueta cor
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultamos (true) o mostramos (false) el nombre del objeto gráfico en la lista de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- establecemos la prioridad para obtener el evento de cliquear sobre el gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mueve la etiqueta rectangular                                    |
//+------------------------------------------------------------------+
bool RectLabelMove(const long   chart_ID=0,       // ID del gráfico
                   const string name="RectLabel", // nombre de la etiqueta
                   const int    x=0,              // coordenada por el eje X
                   const int    y=0)              // coordenada por el eje Y
  {
//--- anulamos el valor del error
   ResetLastError();
//--- movemos la etiqueta rectangular
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada X de la etiqueta! Código del error = ",Ge
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al mover la coordenada Y de la etiqueta! Código del error = ",Ge
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el tamaño de la etiqueta rectangular                      |
//+------------------------------------------------------------------+
bool RectLabelChangeSize(const long   chart_ID=0,       // ID del gráfico
                         const string name="RectLabel", // nombre de la etiqueta
                         const int    width=50,         // ancho de la etiqueta
                         const int    height=18)        // alto de la etiqueta
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos las dimensiones de la etiqueta
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el ancho de la etiqueta! Código del error = ",GetLast
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el alto de la etiqueta! Código del error = ",GetLastE
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Cambia el tipo del borde de la etiqueta rectangular              |
//+------------------------------------------------------------------+
bool RectLabelChangeBorderType(const long             chart_ID=0,           // ID del
                               const string           name="RectLabel",     // nombre
                               const ENUM_BORDER_TYPE border=BORDER_SUNKEN) // tipo de
  {
//--- anulamos el valor del error
   ResetLastError();
//--- cambiamos el tipo del borde
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border))
     {
      Print(__FUNCTION__,
            ": ¡Fallo al cambiar el tipo del borde! Código del error = ",GetLastError
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina la etiqueta rectangular                                  |
//+------------------------------------------------------------------+
bool RectLabelDelete(const long   chart_ID=0,       // ID del gráfico
                     const string name="RectLabel") // nombre de la etiqueta
  {
//--- anulamos el valor del error
   ResetLastError();
//--- eliminamos la etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": ¡Fallo al eliminar la etiqueta rectangular! Código del error = ",GetLas
      return(false);
     }
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tamaño de la ventana del gráfico
   long x_distance;
   long y_distance;
//--- definimos las dimensiones de la ventana
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("¡Fallo al obtener el ancho del gráfico! Código del error = ",GetLastError
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("¡Fallo al obtener el alto del gráfico! Código del error = ",GetLastError
      return;
     }
//--- definimos las coordenadas de la etiqueta rectangular
   int x=(int)x_distance/4;
   int y=(int)y_distance/4;
//--- establecemos las dimensiones de la etiqueta
   int width=(int)x_distance/4;
   int height=(int)y_distance/4;
//--- creamos la etiqueta rectangular
   if(!RectLabelCreate(0,InpName,0,x,y,width,height,InpBackColor,InpBorder,InpCorner,
      InpColor,InpStyle,InpLineWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- cambiamos el tamaño de la etiqueta rectangular
   int steps=(int)MathMin(x_distance/4,y_distance/4);
   for(int i=0;i<steps;i++)
     {
      //--- cambiamos el tamaño
      width+=1;
      height+=1;
      if(!RectLabelChangeSize(0,InpName,width,height))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- comprobamos si el trabajo del script ha sido finalizado forzosamente
      if(IsStopped())
         return;
      //--- redibujamos el gráfico y esperamos 0,01 segundo
      ChartRedraw();
      Sleep(10);
     }
//--- retardo de 1 segundo
   Sleep(1000);
//--- cambiamos el tipo del borde
   if(!RectLabelChangeBorderType(0,InpName,BORDER_RAISED))
      return;
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- cambiamos el tipo del borde
   if(!RectLabelChangeBorderType(0,InpName,BORDER_SUNKEN))
      return;
//--- redibujamos el gráfico y esperamos 1 segundo
   ChartRedraw();
   Sleep(1000);
//--- eliminamos la etiqueta
   RectLabelDelete(0,InpName);
   ChartRedraw();
//--- esperamos 1 segundo
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

Propiedades de objetos

Los objetos gráficos pueden tener muchas propiedades, dependiendo del tipo del objeto. Para establecer y obtener los valores de las propiedades de los objetos se utilizan las correspondientes funciones de trabajo con objetos gráficos.

Todos los objetos utilizados en el análisis técnico están enlazados con los gráficos conforme a las coordenadas de precio y tiempo, la línea de tendencia, los canales, los instrumentos de Fibonacci, etc. Pero hay una serie de objetos auxiliares pensados para mejorar el interfaz, y que tienen un enlace con una parte siempre visible del gráfico (la ventana principal del gráfico o las subventanas de los indicadores):

Objeto	Identificador	X/Y	Width/ Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	—	—	Sí	—	Sí	Sí
Label	OBJ_LABEL	Sí	Sí (sólo lectura)	—	Sí	Sí	Sí
Button	OBJ_BUTTON	Sí	Sí	—	Sí	—	—
Bitmap	OBJ_BITMAP	—	Sí (sólo lectura)	Sí	—	Sí	—
Bitmap Label	OBJ_BITMAP_LABEL	Sí	Sí (sólo lectura)	—	Sí	Sí	—
Edit	OBJ_EDIT	Sí	Sí	—	Sí	—	—
Rectangle Label	OBJ_RECTANGLE_LABEL	Sí	Sí	—	Sí	—	—

En el recuadro se usan las siguientes denominaciones:

- **X/Y** - las coordenadas del punto de enlace se establecen en píxeles con respecto a uno de los ángulos del gráfico;
- **Width/Height** - el objeto tiene anchura y altura. Si se indica "sólo lectura", esto significa que los valores de la anchura y la altura se calculan sólo después de que el objeto haya sido dibujado en el gráfico;
- **Date/Price** - las coordenadas del punto de enlace se dan con la pareja fecha/hora;
- **OBJPROP_CORNER** - establece el ángulo del gráfico con respecto al cual se indican las coordenadas del punto de enlace. Puede ser uno de los 4 valores de la enumeración ENUM_BASE_CORNER;
- **OBJPROP_ANCHOR** - establece la posición del punto de enlace en el propio objeto, y puede ser uno de los 9 valores de la enumeración ENUM_ANCHOR_POINT. Precisamente desde este punto hasta el ángulo elegido en el gráfico se indican las coordenadas en píxeles;
- **OBJPROP_ANGLE** - establece el ángulo de rotación del objeto en el sentido opuesto a las agujas del reloj.

© 2000-2025, MetaQuotes Ltd.

Las funciones que establecen las propiedades de objetos gráficos, así como las operaciones de creación ObjectCreate() y movimiento ObjectMove() de los objetos en el gráfico, sirven prácticamente para mandar los comandos al gráfico. Cuando estas funciones se ejecuten con éxito, el comando se coloca en la cola general de los eventos del gráfico. El cambio visual de los objetos gráficos se realiza durante el procesamiento de la cola de eventos de este gráfico.

Por esta razón no hay que esperar la modificación visual inmediata de los objetos gráficos tras la llamada de estas funciones. En general la actualización de los objetos gráficos se realiza por el terminal de forma automática según los eventos del cambio, es decir: la llegada de una nueva cotización, cambio del tamaño de la ventana, etc.

Para la actualización forzosa de los objetos gráficos, se utiliza el comando de redibujo del gráfico ChartRedraw().

Para las funciones ObjectSetInteger() y ObjectGetInteger()

ENUM_OBJECT_PROPERTY_INTEGER

Identificador	Descripción	Tipo de la propiedad
OBJPROP_COLOR	Color	color
OBJPROP_STYLE	Estilo	ENUM_LINE_STYLE
OBJPROP_WIDTH	Grosor de línea	int
OBJPROP_BACK	Objeto en el fondo	bool
OBJPROP_ZORDER	La prioridad de un objeto gráfico para obt	long

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	ener el evento de cliquear sobre el gráfico (CHARTEVENT_CLICK). Por defecto, cuando se crea un objeto, este valor se pone a cero; pero si hace	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	falta, se puede subir la prioridad. Cuando los objetos se aplican uno al otro, sólo uno de ellos, cuya prioridad es superior, recibirá el evento CHAR	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	TEVENT_CLICK.	
OBJPROP_FILL	Relleno de objeto con color (para OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESS	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	ION)	
OBJPROP_HIDDEN	Prohíbe mostrar el nombre del objeto gráfico en la lista de objetos del menú del terminal "Gráficos" - "Objetos" - "Lista de objetos". El	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	valor true permite ocultar el objeto que el usuario no necesita. Por defecto, true se pone para los objetos que muestran los eventos del calend	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	ario, historial de trading, así como para los objetos creados en un programa MQL5. Para poder ver estos objetos gráfico y acceder a sus propie	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	dades, hay que pulsar en el botón "Todos" en la ventana "Lista de objetos".	
OBJPROP_SELECTED	Selección de objeto	bool
OBJPROP_READONLY	Posibilidad de editar el texto en el objeto Edit	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
OBJPROP_TYPE	Tipo de objeto	ENUM_OBJECT r/o
OBJPROP_TIME	Coordenadas de tiempo	datetime modificador=número del punto de anclaje
OBJPROP_SELECTABLE	Disponibilidad de objeto	bool
OBJPROP_CREATETIME	Tiempo de creación de objeto	datetime r/o
OBJPROP_LEVELS	Número de niveles	int
OBJPROP_LEVELCOLOR	Color de línea-nivel	color modificador=número del nivel

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
OBJPROP_LEVELSTYLE	Estilo de línea-nivel	ENUM_LINE_STYLE modificador=número del nivel
OBJPROP_LEVELWIDTH	Grosor de línea-nivel	int modificador=número del nivel
OBJPROP_ALIGN	Alineación del texto horizontalmente en el objeto "Campo de edición" (OBJ_EDIT)	ENUM_ALIGN_MODE
OBJPROP_FONTSIZE	Tamaño de caract	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	eres	
OBJPROP_RAY_LEFT	Rayo va a la izquierda	bool
OBJPROP_RAY_RIGHT	Rayo va a la derecha	bool
OBJPROP_RAY	Línea vertical va a través de todas las ventanas del gráfico	bool
OBJPROP_ELLIPSE	Visualización del elipse entero para el	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	objeto "Arcos de Fibonacci" (OBJ_FIBOARC)	
OBJPROP_ARROWCODE	Código de flecha para el objeto "Flecha"	uchar
OBJPROP_TIMEFRAMES	Visibilidad de objeto en períodos de tiempo	juego de banderas flags
OBJPROP_ANCHOR	Posición del punto	ENUM_ARROW_ANCHOR (para OBJ_ARROW), ENUM_ANCHOR_POINT (para OBJ_LABEL, OBJ_BITMAP_LABEL y OBJ_TEXT)

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	de anclaje de un objeto gráfico	
OBJPROP_XDISTANCE	Distancia en píxeles por el eje X desde la esquina de enlace (ver nota)	int
OBJPROP_YDISTANCE	Distancia en píxeles por el eje Y desde	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	la esquina de enlace (ver nota)	
OBJPROP_DIRECTION	Tendencia del objeto de Gann	ENUM_GANN_DIRECTION
OBJPROP_DEGREE	Nivel de marcación ondulada de Elliott	ENUM_ELLIOT_WAVE_DEGREE
OBJPROP_DRAWLINES	Visualización de líneas para la marcació	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	nondulada de Elliott	
OBJPROP_STATE	Estado del botón (Pulsado/Despulsado)	bool
OBJPROP_CHART_ID	Identificador del objeto "Gráfico" (OBJ_CHART). Permite trabajar con las propiedades	long r/o

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	de este objeto como con cualquier otro gráfico usando las funciones descritas en el apartado Operaciones con gráficos, pero hay algunas excepciones.	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
OBJPROP_XSIZE	Ancho del objeto por el eje X en píxeles. Se establece para los objetos OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LAB	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	EL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_YSIZE	Alto del objeto por el eje Y en píxeles. Se establece para los objetos OBJ_LABEL (read only), OBJ_BUTTON, OBJ	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_XOFFSET	Coordenada X de la esquina superior izquierda del área rectangular de	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	visibilidad en los objetos gráficos "Etiqueta gráfica" y "Dibujo" (OBJ_BITMAP_LABEL y OBJ_BITMAP). El valor se establece en píxeles respecto a la esq	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	uina superiorizquierdadelaimagenoriginal.	
OBJPROP_YOFFSET	Coordenada Y de la esquina superiorizquierdadel área rectangular de visibilidad en los objetos grá	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	ficos "Etiqueta gráfica" y "Dibujo" (OBJ_BITMAP_LABEL y OBJ_BITMAP). El valor se establece en píxeles respecto a la esquina superior izquierda de	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	la imagen original.	
OBJPROP_PERIOD	Período de tiempo para el objeto "Gráfico"	ENUM_TIMEFRAMES
OBJPROP_DATE_SCALE	Visualiza la escala de tiempo para el objeto "Gráfico"	bool
OBJPROP_PRICE_SCALE	Visualiza la escala de pre	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	cios para el objeto "Gráfico"	
OBJPROP_CHART_SCALE	Escala para el objeto "Gráfico"	int valores en el rango de 0-5
OBJPROP_BGCOLOR	Color del fondo para el OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	color
OBJPROP_CORNER	Esquina del	ENUM_BASE_CORNER

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	gráfico para enlazar un objeto gráfico	
OBJPROP_BORDER_TYPE	Estilo del borde del objeto "Etiqueta rectangular"	ENUM_BORDER_TYPE
OBJPROP_BORDER_COLOR	Color del borde para el objeto OBJ_EDIT y OBJ_BUTTON	color

Cuando se utilizan las operaciones con gráficos para el objeto "Gráfico" (OBJ_CHART), se procede según las siguientes limitaciones:

© 2000-2025, MetaQuotes Ltd.

- no se puede cerrar usando ChartClose();
- no se puede cambiar símbolo/período usando la función ChartSetSymbolPeriod();
- no funcionan las propiedades CHART_SCALE, CHART_BRING_TO_TOP, CHART_SHOW_DATE_SCALE y CHART_SHOW_PRICE_SCALE (ENUM_CHART_PROPERTY_INTEGER).

Para los objetos OBJ_BITMAP_LABEL y OBJ_BITMAP se puede establecer programáticamente el modo especial de visualización de las imágenes. En este modo se muestra sólo aquella parte de la imagen original a la que se aplica el área rectangular de visibilidad, mientras que el resto de la imagen se hace invisible. El tamaño del área de visibilidad se establece mediante las propiedades OBJPROP_XSIZE y OBJPROP_YSIZE. Usted puede "mover" el área de visibilidad sólo dentro de los márgenes de la imagen original, utilizando las propiedades OBJPROP_XOFFSET y OBJPROP_YOFFSET.

Para los objetos con tamaños fijos: OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT y OBJ_CHART, las propiedades OBJPROP_XDISTANCE y OBJPROP_YDISTANCE establecen la posición del punto superior izquierdo respecto al ángulo del gráfico (OBJPROP_CORNER) del que van a calcularse las coordenadas X y Y en píxeles.

Para las funciones ObjectSetDouble() y ObjectGetDouble()

ENUM_OBJECT_PROPERTY_DOUBLE

Identificador	Descripción	Tipo de la propiedad
OBJPROP_PRICE	Coordenadas de precio	double modificador=número del punto de anclaje
OBJPROP_LEVELVALUE	Valor de nivel	double modificador=número del nivel
OBJPROP_SCALE	Escala (propiedad de obj	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	etos de Gann del objeto "Arcos de Fibonacci")	
OBJPROP_ANGLE	Ángulo. Para los objetos con el ángulo no especificado, creados desde el programa, el valor es igu	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	al a EMPTY_VALUE	
OBJPROP_DEVIATION	Desviación para el canal de la desviación estándar	double

Para las funciones ObjectSetString() y ObjectGetString()

ENUM_OBJECT_PROPERTY_STRING

Identificador	Descripción	Tipo de la propiedad
OBJPROP_NAME	Nombre de objeto	string
OBJPROP_TEXT	Descripción de objeto (te	string

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	xto contenido en el objeto)	
OBJPROP_TOOLTIP	El texto de la ayuda emergente. Si la propiedad no está establecida, entonces sale la ayuda generada automáticament	string

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	e por el terminal. Puede desactivar la visualización de ayuda emergente poniendo el valor "\n" (salto de línea) para esta propiedad	
OBJPROP_LEVELTEXT	Descripción de	string modificador=número del nivel

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	nivel	
OBJPROP_FONT	Fuente	string
OBJPROP_BMPFILE	Nombre del archivo BMP para el objeto "Etiqueta gráfica". Véase también Recursos	string modificador: 0-estado ON, 1-estado OFF
OBJPROP_SYMBOL	Símbolo para el objeto "Gráfico"	string

Para los objetos OBJ_RECTANGLE_LABEL ("Etiqueta rectangular") se puede especificar uno de los tres estilos del borde a los que corresponden los valores de la enumeración ENUM_BORDER_TYPE.

© 2000-2025, MetaQuotes Ltd.

ENUM_BORDER_TYPE

Identificador	Descripción
BORDER_FLAT	Borde plano
BORDER_RAISED	Convexo
BORDER_SUNKEN	Cóncavo

Para el objeto OBJ_EDIT ("Campo de edición") y para la función ChartScreenShot() se puede indicar el tipo de alineación por la horizontal utilizando los valores de la enumeración ENUM_ALIGN_MODE.

ENUM_ALIGN_MODE

Identificador	Descripción
ALIGN_LEFT	Alineación por la izquierda
ALIGN_CENTER	Alineación centrada (sólo para el objeto "Campo de edición")
ALIGN_RIGHT	Alineación derecha

Ejemplo:

```
#define  UP          "\x0431"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="my_OBJ_LABEL_object";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Object ",label_name," not found. Error code = ",GetLastError());
      //--- creamos el objeto Label
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- establecemos la coordenada X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- establecemos la coordenada Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      //--- definimos el color del texto
      ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
      //--- definimos el texto para el objeto Label
      ObjectSetString(0,label_name,OBJPROP_TEXT,UP);
      //--- definimos la fuente
      ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");
      //--- definimos el tamaño de la fuente
      ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- rotamos a 45 grados en el sentido de las agujas del reloj
      ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);
      //--- deshabilitamos la selección del objeto con el ratón
      ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);
      //--- dibujamos todo eso en el gráfico
      ChartRedraw(0);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Modos de enlace de objetos

Los objetos gráficos Text, Label, Bitmap y Bitmap Label (OBJ_TEXT, OBJ_LABEL, OBJ_BITMAP y OBJ_BITMAP_LABEL) pueden tener uno de los 9 métodos diferentes de enlace de sus coordenadas, establecidas por la propiedad OBJPROP_ANCHOR.

Objeto	Identificador	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	—	—	Sí	—	**Sí**	Sí
Label	OBJ_LABEL	Sí	Sí (sólo lectura)	—	Sí	**Sí**	Sí
Button	OBJ_BUTTON	Sí	Sí	—	Sí	—	—
Bitmap	OBJ_BITMAP	—	Sí (sólo lectura)	Sí	—	**Sí**	—
Bitmap Label	OBJ_BITMAP_LABEL	Sí	Sí (sólo lectura)	—	Sí	**Sí**	—
Edit	OBJ_EDIT	Sí	Sí	—	Sí	—	—
Rectangle Label	OBJ_RECTANGLE_LABEL	Sí	Sí	—	Sí	—	—

En el recuadro se usan las siguientes denominaciones:

- **X/Y** - las coordenadas del punto de enlace se establecen en píxeles con respecto a uno de los ángulos del gráfico;
- **Width/Height** - el objeto tiene anchura y altura. Si se indica "sólo lectura", esto significa que los valores de la anchura y la altura se calculan sólo después de que el objeto haya sido dibujado en el gráfico;
- **Date/Price** - las coordenadas del punto de enlace se dan con la pareja fecha/hora;
- **OBJPROP_CORNER** - establece el ángulo del gráfico con respecto al cual se indican las coordenadas del punto de enlace. Puede ser uno de los 4 valores de la enumeración ENUM_BASE_CORNER;
- **OBJPROP_ANCHOR** - establece la posición del punto de enlace en el propio objeto, y puede ser uno de los 9 valores de la enumeración ENUM_ANCHOR_POINT. Precisamente desde este punto hasta el ángulo elegido en el gráfico se indican las coordenadas en píxeles;
- **OBJPROP_ANGLE** - establece el ángulo de rotación del objeto en el sentido opuesto a las agujas del reloj.

Se puede definir una opción necesaria usando la función ObjectSetInteger(handle_de_gráfico, nombre_de_objeto, OBJPROP_ANCHOR, modo_de_enlace), donde modo_de_enlace es uno de los valores de la enumeración ENUM_ANCHOR_POINT.

ENUM_ANCHOR_POINT

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
ANCHOR_LEFT_UPPER	Punto de enlace en la esquina superior izquierda
ANCHOR_LEFT	Punto de enlace a la izquierda en el centro
ANCHOR_LEFT_LOWER	Punto de enlace en la esquina inferior izquierda
ANCHOR_LOWER	Punto de enlace abajo en el centro
ANCHOR_RIGHT_LOWER	Punto de enlace en la esquina inferior derecha
ANCHOR_RIGHT	Punto de enlace a la derecha en el centro
ANCHOR_RIGHT_UPPER	Punto de enlace en la esquina superior derecha
ANCHOR_UPPER	Punto de enlace arriba en el centro
ANCHOR_CENTER	Punto de enlace justo en el centro del objeto

Los objetos OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT y OBJ_CHART tienen el punto de anclaje fijo en la esquina superior izquierda (ANCHOR_LEFT_UPPER).

Ejemplo:

```
string text_name="my_OBJ_TEXT_object";
if(ObjectFind(0,text_name)<0)
  {
   Print("Object ",text_name," not found. Error code = ",GetLastError());
   //--- obtenemos el precio máximo del gráfico
   double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
   //--- creamos el objeto Label
   ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
   //--- definimos el color del texto
   ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
   //--- definimos el color del fondo
   ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
   //--- definimos el texto para el objeto Label
   ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
   //--- definimos la fuente
   ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
   //--- definimos el tamaño la fuente
   ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
   //--- enlazamos a la esquina superior derecha
   ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
   //--- rotamos a 90 grados contra el sentido de reloj
   ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
   //--- prohibimos la selección del objeto con el ratón
   ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
   //--- lo dibujamos en el gráfico
   ChartRedraw(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

Los objetos gráficos Arrow (OBJ_ARROW) tienen sólo 2 opciones de enlazar sus coordinadas. Los identificadores se enumeran en ENUM_ARROW_ANCHOR.

ENUM_ARROW_ANCHOR

Identificador	Descripción
ANCHOR_TOP	Punto de enlace para la flecha se encuentra arriba
ANCHOR_BOTTOM	Punto de enlace para la flecha se encuentra abajo

Ejemplo:

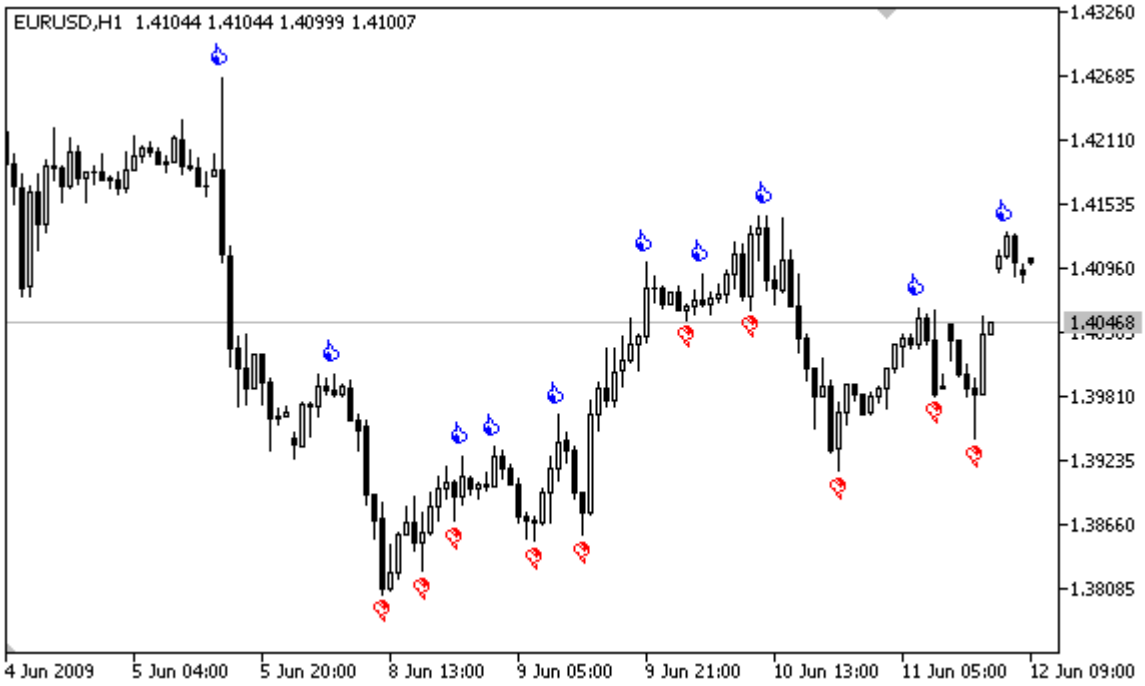
```
void OnStart()
  {
//--- arrays auxiliares
   double Ups[],Downs[];
   datetime Time[];
//--- definimos los arrays como series temporales
   ArraySetAsSeries(Ups,true);
   ArraySetAsSeries(Downs,true);
   ArraySetAsSeries(Time,true);
//--- creamos el manejador del indicador Fractals
   int FractalsHandle=iFractals(NULL,0);
   Print("FractalsHandle = ",FractalsHandle);
//--- ponemos el último error a cero
   ResetLastError();
//--- intentamos copiar valores del indicador
   int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Print("Fallo al copiar los fractales superiores. Error = ",GetLastError());
      return;
     }

   ResetLastError();
//--- intentamos copiar valores del indicador
   copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
   if(copied<=0)
     {
      Print("Fallo al copiar los fractales inferiores. Error = ",GetLastError());
      return;
     }

   ResetLastError();
//--- copiamos la serie temporal que contiene la hora de apertura de las últimas 1000
   copied=CopyTime(NULL,0,0,1000,Time);
   if(copied<=0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("Fallo al copiar la hora de apertura de los últimos 1000 barras");
      return;
     }

   int upcounter=0,downcounter=0; // vamos a contar el número de flechas en ellas
   bool created;// vamos a recibir el resultado del intento de creación del objeto
   for(int i=2;i<copied;i++)// repasamos los valores del indicador iFractals
     {
      if(Ups[i]!=EMPTY_VALUE)// encontramos el fractal de arriba
        {
         if(upcounter<10)// creamos no más de 10 objetos "arriba"
           {
            //--- intentamos crear el objeto "arriba"
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[i]
            if(created)// si ha salido, vamos a tunearlo
              {
               //--- punto de enlace abajo para no pisar la barra
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
               //--- el último retoque - pintamos
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
               upcounter++;
              }
           }
        }
      if(Downs[i]!=EMPTY_VALUE)// encontramos el fractal de abajo
        {
         if(downcounter<10)// creamos no más de 10 objetos "abajo"
           {
            //--- intentamos crear el objeto "abajo"
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Dowr
            if(created)// si lo conseguimos, vamos a tunearlo
              {
               //--- punto de enlace arriba para no pisar la barra
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
               //--- el último retoque - pintamos
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
               downcounter++;
              }
           }
        }
     }
  }
```

Una vez completado el script, el gráfico será más o menos como se muestra abajo.

© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

Esquina del gráfico a la que se enlaza un objeto

Existe una serie de objetos gráficos, para los que se puede establecer el ángulo del gráfico con respecto al cual se indican las coordenadas en píxeles. Se trata de los siguientes tipos de objetos (entre paréntesis se indican los identificadores del tipo de objeto):

- Label (OBJ_LABEL); Etiqueta de texto
- Button (OBJ_BUTTON); Botón
- Bitmap Label (OBJ_BITMAP_LABEL); Etiqueta Gráfica
- Edit (OBJ_EDIT); Campo de edición
- Rectangle Label (OBJ_RECTANGLE_LABEL). Etiqueta rectangular

Objeto	Identificador	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	—	—	Sí	—	Sí	Sí
Label	OBJ_LABEL	Sí	Sí (sólo lectura)	—	**Sí**	Sí	Sí
Button	OBJ_BUTTON	Sí	Sí	—	**Sí**	—	—
Bitmap	OBJ_BITMAP	—	Sí (sólo lectura)	Sí	—	Sí	—
Bitmap Label	OBJ_BITMAP_LABEL	Sí	Sí (sólo lectura)	—	**Sí**	Sí	—
Edit	OBJ_EDIT	Sí	Sí	—	**Sí**	—	—
Rectangle Label	OBJ_RECTANGLE_LABEL	Sí	Sí	—	**Sí**	—	—

En el recuadro se usan las siguientes denominaciones:

- **X/Y** - las coordenadas del punto de enlace se establecen en píxeles con respecto a uno de los ángulos del gráfico;
- **Width/Height** - el objeto tiene anchura y altura. Si se indica "sólo lectura", esto significa que los valores de la anchura y la altura se calculan sólo después de que el objeto haya sido dibujado en el gráfico;
- **Date/Price** - las coordenadas del punto de enlace se dan con la pareja fecha/hora;
- **OBJPROP_CORNER** - establece el ángulo del gráfico con respecto al cual se indican las coordenadas del punto de enlace. Puede ser uno de los 4 valores de la enumeración ENUM_BASE_CORNER;
- **OBJPROP_ANCHOR** - establece la posición del punto de enlace en el propio objeto, y puede ser uno de los 9 valores de la enumeración ENUM_ANCHOR_POINT. Precisamente desde este punto hasta el ángulo elegido en el gráfico se indican las coordenadas en píxeles;

© 2000-2025, MetaQuotes Ltd.

- **OBJPROP_ANGLE** - establece el ángulo de rotación del objeto en el sentido opuesto a las agujas del reloj.

Para especificar la esquina del gráfico desde la cual se miden las coordinadas X y Y en píxeles, hay que usar la función ObjectSetInteger(chartID, name, OBJPROP_CORNER, **chart_corner**), donde:

- chartID - identificador del gráfico;
- name - nombre del objeto gráfico;
- OBJPROP_CORNER - identificador de la propiedad para determinar la esquina de enlace;
- **chart_corner** - la esquina del gráfico puede adquirir uno de los valores de la enumeración ENUM_BASE_CORNER.

ENUM_BASE_CORNER

Identificador	Descripción
CORNER_LEFT_UPPER	Centro de coordinadas se encuentra en la esquina superior izquierda del gráfico
CORNER_LEFT_LOWER	Centro de coordinadas se encuentra en la esquina inferior izquierda del gráfico
CORNER_RIGHT_LOWER	Centro de coordinadas se encuentra en la esquina inferior derecha del gráfico
CORNER_RIGHT_UPPER	Centro de coordinadas se encuentra en la esquina superior derecha del gráfico

Ejemplo:

```
void CreateLabel(long   chart_id,
                 string name,
                 int    chart_corner,
                 int    anchor_point,
                 string text_label,
                 int    x_ord,
                 int    y_ord)
  {
//---
   if(ObjectCreate(chart_id,name,OBJ_LABEL,0,0,0))
     {
      ObjectSetInteger(chart_id,name,OBJPROP_CORNER,chart_corner);
      ObjectSetInteger(chart_id,name,OBJPROP_ANCHOR,anchor_point);
      ObjectSetInteger(chart_id,name,OBJPROP_XDISTANCE,x_ord);
      ObjectSetInteger(chart_id,name,OBJPROP_YDISTANCE,y_ord);
      ObjectSetString(chart_id,name,OBJPROP_TEXT,text_label);
     }
   else
      Print("No se ha logrado crear el objeto OBJ_LABEL ",name,", Código de error = ",
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int height=(int)ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=(int)ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   string arrows[4]={"LEFT_UPPER","RIGHT_UPPER","RIGHT_LOWER","LEFT_LOWER"};
   CreateLabel(0,arrows[0],CORNER_LEFT_UPPER,ANCHOR_LEFT_UPPER,arrows[0],50,50);
   CreateLabel(0,arrows[1],CORNER_RIGHT_UPPER,ANCHOR_RIGHT_UPPER,arrows[1],50,50);
   CreateLabel(0,arrows[2],CORNER_RIGHT_LOWER,ANCHOR_RIGHT_LOWER,arrows[2],50,50);
   CreateLabel(0,arrows[3],CORNER_LEFT_LOWER,ANCHOR_LEFT_LOWER,arrows[3],50,50);
  }
```

© 2000-2025, MetaQuotes Ltd.

Visibilidad de objetos

La combinación de banderas de la visibilidad de objeto determina los períodos de tiempo del gráfico en los que se muestra el objeto. Para definir/obtener los valores de la propiedad OBJPROP_TIMEFRAMES podemos usar las funciones ObjectSetInteger()/ObjectGetInteger().

Constante	Valor	Descripción
OBJ_NO_PERIODS	0	El objeto no se muestra en ninguno de los períodos
OBJ_PERIOD_M1	0x00000001	El objeto se dibuja en los gráficos de 1 minuto
OBJ_PERIOD_M2	0x00000002	El objeto se dibuja en los gráficos de 2 minutos
OBJ_PERIOD_M3	0x00000004	El objeto se dibuja en los gráficos de 3 minutos
OBJ_PERIOD_M4	0x00000008	El objeto se dibuja en los gráficos de 4 minutos
OBJ_PERIOD_M5	0x00000010	El objeto se dibuja en los gráficos de 5 minutos
OBJ_PERIOD_M6	0x00000020	El objeto se dibuja en los gráficos de 6 minutos
OBJ_PERIOD_M10	0x00000040	El objeto se dibuja en los gráficos de 10 minutos
OBJ_PERIOD_M12	0x00000080	El objeto se dibuja en los gráficos de 12 minutos
OBJ_PERIOD_M15	0x00000100	El objeto se dibuja en los gráficos de 15 minutos
OBJ_PERIOD_M20	0x00000200	El objeto se dibuja en los gráficos de 20 minutos
OBJ_PERIOD_M30	0x00000400	El objeto se dibuja en los gráficos de 30 minutos
OBJ_PERIOD_H1	0x00000800	El objeto se dibuja en los gráficos de 1 hora
OBJ_PERIOD_H2	0x00001000	El objeto se dibuja en los gráficos de 2 horas
OBJ_PERIOD_H3	0x00002000	El objeto se dibuja en los gráficos de 3 horas
OBJ_PERIOD_H4	0x00004000	El objeto se dibuja en los gráficos de 4 horas

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
OBJ_PERIOD_H6	0x00008000	El objeto se dibuja en los gráficos de 6 horas
OBJ_PERIOD_H8	0x00010000	El objeto se dibuja en los gráficos de 8 horas
OBJ_PERIOD_H12	0x00020000	El objeto se dibuja en los gráficos de 12 horas
OBJ_PERIOD_D1	0x00040000	El objeto se dibuja en los gráficos de un día
OBJ_PERIOD_W1	0x00080000	El objeto se dibuja en los gráficos semanales
OBJ_PERIOD_MN1	0x00100000	El objeto se dibuja en los gráficos mensuales
OBJ_ALL_PERIODS	0x001fffff	El objeto se dibuja en todos los períodos de tiempo

Las banderas de la visibilidad se puede combinar mediante el símbolo "|", por ejemplo, la combinación de banderas OBJ_PERIOD_M10|OBJ_PERIOD_H4 significa que el objeto será mostrado en los períodos de 10 minutos y 4 horas.

Ejemplo:

```
void OnStart()
  {
//---
   string highlevel="PreviousDayHigh";
   string lowlevel="PreviousDayLow";
   double prevHigh;           // High del día anterior
   double prevLow;            // Low del día anterior
   double highs[],lows[];     // matrices para recibir High y Low

//--- ponemos el último error a cero
   ResetLastError();
//--- obtenemos 2 últimos valores High en período de tiempo diurno
   int highsgot=CopyHigh(Symbol(),PERIOD_D1,0,2,highs);
   if(highsgot>0) // si el copiado ha sido con éxito
     {
      Print("Precios High de los últimos 2 días han sido recibidos con éxito");
      prevHigh=highs[0]; // High del día anterior
      Print("prevHigh = ",prevHigh);
      if(ObjectFind(0,highlevel)<0) // objeto con el nombre highlevel no ha sido encor
        {
         ObjectCreate(0,highlevel,OBJ_HLINE,0,0,0); // creamos el objeto Línea Horizor
        }
      //--- definimos el nivel de precio para la línea highlevel
      ObjectSetDouble(0,highlevel,OBJPROP_PRICE,0,prevHigh);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- establecemos la visibilidad sólo para PERIOD_M10 y PERIOD_H4
      ObjectSetInteger(0,highlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else
     {
      Print("Fallo al recibir los precios High de los últimos 2 días, Error = ",GetLas
     }

//--- ponemos el último error a cero
   ResetLastError();
//--- obtenemos 2 últimos valores Low en período de tiempo diurno
   int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2,lows);
   if(lowsgot>0) // si el copiado ha sido con éxito
     {
      Print("Precios Low de los últimos 2 días han sido recibidos con éxito");
      prevLow=lows[0]; // Low del día anterior
      Print("prevLow = ",prevLow);
      if(ObjectFind(0,lowlevel)<0) // objeto con el nombre lowlevel no ha sido enconti
        {
         ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // creamos el objeto Línea Horizont
        }
      //--- definimos el nivel de precio para la línea lowlevel
      ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
      //--- establecemos la visibilidad sólo para  PERIOD_M10 y PERIOD_H4
      ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else Print("Fallo al recibir los precios Low de los últimos 2 días, Error = ",GetLa

   ChartRedraw(0); // redibujamos el gráfico por vía forzada
  }
```

Véase también

PeriodSeconds, Period, Períodos de gráficos, Fecha y hora

© 2000-2025, MetaQuotes Ltd.

Niveles de las ondas de Elliott

Las ondas de Elliott están representadas por dos objetos gráficos de los tipos OBJ_ELLIOTWAVE5 y OBJ_ELLIOTWAVE3. Para definir el tamaño de la onda (método de etiquetar las ondas) se usa la propiedad OBJPROP_DEGREE, a la que podemos asignar uno de los valores de la enumeración ENUM_ELLIOT_WAVE_DEGREE.

ENUM_ELLIOT_WAVE_DEGREE

Constante	Descripción
ELLIOTT_GRAND_SUPERCYCLE	Gran superciclo (Grand Supercycle)
ELLIOTT_SUPERCYCLE	Superciclo (Supercycle)
ELLIOTT_CYCLE	Ciclo (Cycle)
ELLIOTT_PRIMARY	Primario (Primary)
ELLIOTT_INTERMEDIATE	Intermedio (Intermediate)
ELLIOTT_MINOR	Menor (Minor)
ELLIOTT_MINUTE	Minute (Minute)
ELLIOTT_MINUETTE	Minuette (Minuette)
ELLIOTT_SUBMINUETTE	Subminuette (Subminuette)

Ejemplo:

```
   for(int i=0;i<ObjectsTotal(0);i++)
     {
      string currobj=ObjectName(0,i);
      if((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE3)  ||
         ((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE5)))
        {
         //--- pongamos el nivel de marcación en INTERMEDIATE
         ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
         //--- habilitamos la muestra de las líneas entre las partes superiores de las
         ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
         //--- definimos el color de las líneas
         ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
         //--- definimos el grosor de las líneas
         ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
         //--- definimos la descripción
         ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

Objetos de Gann

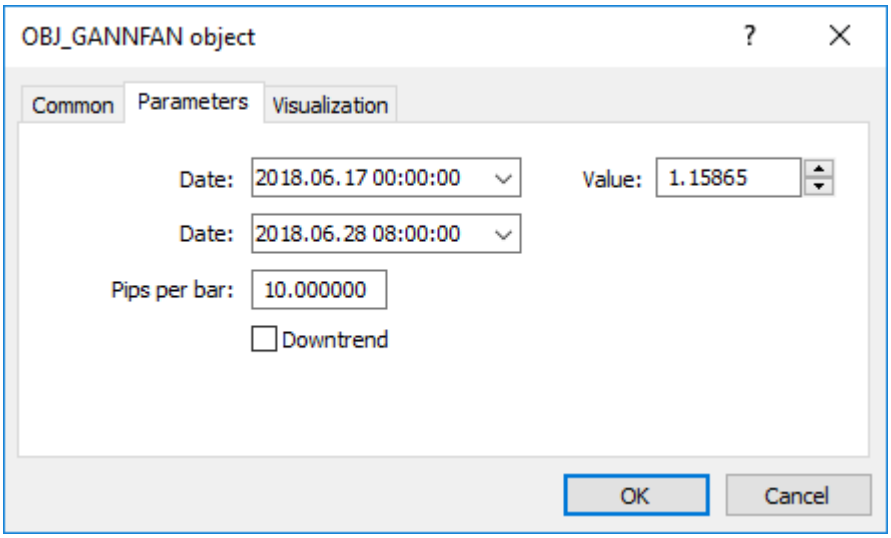
Para los objetos abanico de Gann (OBJ_GANNFAN) y retícula de Gann (OBJ_GANNGRID) podemos especificar uno de dos valores de la enumeración ENUM_GANN_DIRECTION que establece la dirección de tendencia.

ENUM_GANN_DIRECTION

Constante	Descripción
GANN_UP_TREND	Línea de tendencia al alza
GANN_DOWN_TREND	Línea de tendencia a la baja

Para establecer la escala de la línea principar de 1x1 se usa la función ObjectSetDouble(chart_handle, gann_object_name, OBJPROP_SCALE, scale), donde:

- chart_handle - ventana del gráfico en la que se encuentra el objeto;
- gann_object_name - nombre del objeto;
- OBJPROP_SCALE - identificador de la propiedad "Scale";
- scale - escala requerida en unidades Pips/Bar.



Ejemplo de creación del abanico de Gann:

```
void OnStart()
  {
//---
   string my_gann="OBJ_GANNFAN object";
   if(ObjectFind(0,my_gann)<0)// objeto no encontrado
     {
      //--- informamos sobre el fallo
      Print("Object ",my_gann," not found. Error code = ",GetLastError());
      //--- obtenemos el precio máximo del gráfico
      double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
      //--- obtenemos el precio mínimo del gráfico
      double chart_min_price=ChartGetDouble(0,CHART_PRICE_MIN,0);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- ¿Cuántas barras se muestra en el gráfico?
      int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
      //--- creamos una matriz en la que escribimos la hora de apertura de cada barra
      datetime Time[];
      //--- organizamos el acceso a la matriz como en serie temporal
      ArraySetAsSeries(Time,true);
      //--- ahora copiamos en él los datos de barras visibles en el gráfico
      int times=CopyTime(NULL,0,0,bars_on_chart,Time);
      if(times<=0)
        {
         Print("Fallo al copiar un array con la hora de apertura!");
         return;
        }
      //--- preparativos preliminares finalizados

      //--- índice de la barra central en el gráfico
      int center_bar=bars_on_chart/2;
      //--- ecuador del gráfico - entre el máximo y el mínimo
      double mean=(chart_max_price+chart_min_price)/2.0;
      //--- establecemos las coordinadas del primer punto de enlace en el centro
      ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
                   //--- el segundo punto de enlace a la derecha
                   Time[center_bar/2],(mean+chart_min_price)/2.0);
      Print("Time[center_bar] = "+(string)Time[center_bar]+"  Time[center_bar/2] = "+
      //Print("Time[center_bar]/="+Time[center_bar]+"  Time[center_bar/2]="+Time[cente
      //--- establecemos la escala en unidades Pips/Bar
      ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
      //--- definimos la tendencia de la línea
      ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
      //--- definimos el grosor de la línea
      ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
      //--- definimos el estilo de la línea
      ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
      //--- y el color de la línea
      ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
      //--- permitimos al usuario seleccionar el objeto
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
      //--- lo seleccionamos nosotros mismos
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
      //--- redibujamos el gráfico
      ChartRedraw(0);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Colores Web

La siguientes constantes de colores están predefinidas para el tipo color:

clrBlack	clrDarkGreen	clrDarkSlateGray	clrOlive	clrGreen	clrTeal	clrNavy	clrPurple
clrMaroon	clrIndigo	clrMidnightBlue	clrDarkBlue	clrDarkOliveGreen	clrSaddleBrown	clrForestGreen	clrOliveDrab
clrSeaGreen	clrDarkGoldenrod	clrDarkSlateBlue	clrSienna	clrMediumBlue	clrBrown	clrDarkTurquoise	clrDimGray
clrLightSeaGreen	clrDarkViolet	clrFireBrick	clrMediumVioletRed	clrMediumSeaGreen	clrChocolate	clrCrimson	clrSteelBlue
clrGoldenrod	clrMediumSpringGreen	clrLawnGreen	clrCadetBlue	clrDarkOrchid	clrYellowGreen	clrLimeGreen	clrOrangeRed
clrDarkOrange	clrOrange	clrGold	clrYellow	clrChartreuse	clrLime	clrSpringGreen	clrAqua
clrDeepSkyBlue	clrBlue	clrMagenta	clrRed	clrGray	clrSlateGray	clrPeru	clrBlueViolet
clrLightSlateGray	clrDeepPink	clrMediumTurquoise	clrDodgerBlue	clrTurquoise	clrRoyalBlue	clrSlateBlue	clrDarkKhaki
clrIndianRed	clrMediumOrchid	clrGreenYellow	clrMediumAquamarine	clrDarkSeaGreen	clrTomato	clrRosyBrown	clrOrchid
clrMediumPurple	clrPaleVioletRed	clrCoral	clrCornflowerBlue	clrDarkGray	clrSandyBrown	clrMediumSlateBlue	clrTan
clrDarkSalmon	clrBurlyWood	clrHotPink	clrSalmon	clrViolet	clrLightCoral	clrSkyBlue	clrLightSalmon
clrPlum	clrKhaki	clrLightGreen	clrAquamarine	clrSilver	clrLightSkyBlue	clrLightSteelBlue	clrLightBlue
clrPaleGreen	clrThistle	clrPowderBlue	clrPaleGoldenrod	clrPaleTurquoise	clrLightGray	clrWheat	clrNavajoWhite
clrMoccasin	clrLightPink	clrGainsboro	clrPeachPuff	clrPink	clrBisque	clrLightGoldenrod	clrBlanchedAlmond
clrLemonChiffon	clrBeige	clrAntiqueWhite	clrPapayaWhip	clrCornsilk	clrLightYellow	clrLightCyan	clrLinen
clrLavender	clrMistyRose	clrOldLace	clrWhiteSmoke	clrSeashell	clrIvory	clrHoneydew	clrAliceBlue
clrLavenderBlush	clrMintCream	clrSnow	clrWhite				

Podemos definir el color para un objeto usando la función ObjectSetInteger(), y para indicadores personalizados usando la función PlotIndexSetInteger(). Las funciones similares ObjectGetInteger() y PlotIndexGetInteger() sirven para obtener el valor de un color.

© 2000-2025, MetaQuotes Ltd.

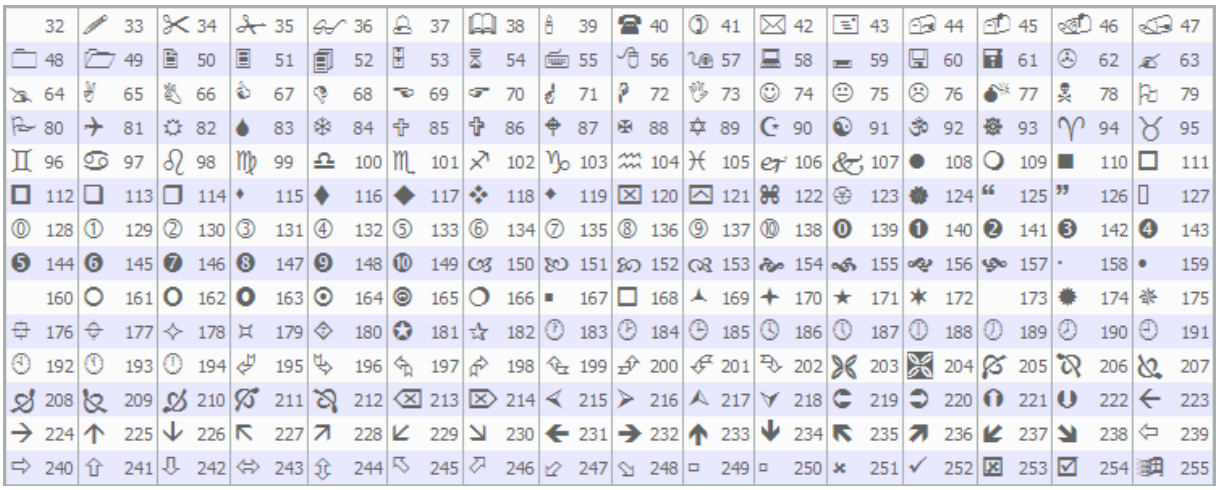
Ejemplo:

```
//---- indicator settings
#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_type3   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_color2  clrRed
#property indicator_color3  clrLime
```

© 2000-2025, MetaQuotes Ltd.

Wingdings

Estos son los caracteres de la fuente Wingdings que se usan con el objeto OBJ_ARROW:



El símbolo necesario se establece usando la función ObjectSetInteger().

Ejemplo:

```
void OnStart()
  {
//---
   string up_arrow="up_arrow";
   datetime time=TimeCurrent();
   double lastClose[1];
   int close=CopyClose(Symbol(),Period(),0,1,lastClose);         // obtenemos el precio
//--- si hemos recibido el precio
   if(close>0)
     {
      ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);           // creamos la flecha
      ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);     // definimos el código de
      ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);         // definimos la hora
      ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]);// definimos el precio
      ChartRedraw(0);                                         // redibujamos la ventana
     }
   else
      Print("Fallo al recibir el último precio Close!");
  }
```

© 2000-2025, MetaQuotes Ltd.

Constantes de indicadores

Hay 37 constantes predefinidas de indicadores técnicos estándares que se puede utilizar en los programas en el lenguaje MQL5. Además, existe la posibilidad de crear sus propios indicadores personalizados mediante la función iCustom(). Todas las constantes necesarias para eso están divididas en 5 grupos:

- Constantes de precio - para seleccionar el tipo de precio o volumen por los cuales se calcula el indicador;
- Métodos de alisamiento - métodos buil-in de alisamiento utilizados en los indicadores;
- Líneas de indicadores - identificadores de buffers de indicadores durante el acceso a los valores de indicadores usando la función CopyBuffer();
- Estilos de dibujo - para especificar uno de los 18 tipos de dibujo y para establecer el estilo de dibujo de la línea;
- Propiedades de indicadores personalizados - se usa en las funciones para trabajar con los indicadores personalizados;
- Tipos de indicadores - sirven para especificar el tipo de indicador técnico a la hora de crear un manejador (handle) a través de la función IndicatorCreate();
- Identificadores de tipos de datos - se usan para establecer el tipo de datos traspasados por el array del tipo MqlParam en la función IndicatorCreate().

© 2000-2025, MetaQuotes Ltd.

Constantes de precios

Los indicadores técnicos requieren para sus cálculos la especificación de los valores de los precios y/o volúmenes a base de los cuales serán calculados. Hay 7 identificadores predefinidos de la enumeración ENUM_APPLIED_PRICE que se usan para especificar la base de precios necesaria para los cálculos.

ENUM_APPLIED_PRICE

Identificador	Descripción
PRICE_CLOSE	Precio de cierre
PRICE_OPEN	Precio de apertura
PRICE_HIGH	Precio máximo en el período
PRICE_LOW	Precio mínimo en el período
PRICE_MEDIAN	Precio mediano, (high+low)/2
PRICE_TYPICAL	Precio típico, (high+low+close)/3
PRICE_WEIGHTED	Precio promedio, (high+low+close+close)/4

Si en los cálculos se usa el volumen, hay que indicar uno de dos valores de la enumeración ENUM_APPLIED_VOLUME.

ENUM_APPLIED_VOLUME

Identificador	Descripción
VOLUME_TICK	Volumen de tick
VOLUME_REAL	Volumen comercial

El indicador técnico iStochastic() se calcula de dos maneras, utilizando:

- o sólo los precios Close;
- o los precios High y Low.

Para eligir la opción necesaria de cálculo hay que indicar uno de los valores de la enumeración ENUM_STO_PRICE.

ENUM_STO_PRICE

Identificador	Descripción
STO_LOWHIGH	Calcular basándose en los precios Low/High
STO_CLOSECLOSE	Calcular basándose en los precios Close/Close

Si un indicador técnico para sus cálculos utiliza los datos cuyo tipo es definido por la enumeración ENUM_APPLIED_PRICE, entonces el manejador de cualquier indicador (built-in en el terminal o escrito por el usuario) puede ser usado como serie de precios de entrada. En este caso para los cálculos serán utilizados los valores del buffer cero del indicador. Eso permite desarrollar fácilmente los valores de un indicador a base de los valores del otro. El manejador de un indicador personalizado se crea llamando a la función iCustom().

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- input parameters
input int      RSIperiod=14;         // período para calcular RSI
input int      Smooth=8;             // período de alisamiento RSI
input ENUM_MA_METHOD meth=MODE_SMMA; // método de alisamiento
//---- plot RSI
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- plot RSI_Smoothed
#property indicator_label2  "RSI_Smoothed"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrNavy
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- indicator buffers
double         RSIBuffer[];          // aquí vamos a almacenar los valores RSI
double         RSI_SmoothedBuffer[]; // aquí estarán los valores suavizados RSI
int            RSIhandle;            // descriptor de indicador RSI
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,RSIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,RSI_SmoothedBuffer,INDICATOR_DATA);
   IndicatorSetString(INDICATOR_SHORTNAME,"iRSI");
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//---
   RSIhandle=iRSI(NULL,0,RSIperiod,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[]
                )

  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ponemos a cero el valor del último error
   ResetLastError();
//--- obtenemos los datos del indicador RSI en un array RSIBuffer[]
   int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);
   if(copied<=0)
     {
      Print("Fallo al copiar los valores del indicador RSI. Error = ",
            GetLastError(),",  copied = ",copied);
      return(0);
     }
//--- creamos el indicador de la media usando los valores del indicador RSI
   int RSI_MA_handle=iMA(NULL,0,Smooth,0,meth,RSIhandle);
   copied=CopyBuffer(RSI_MA_handle,0,0,rates_total,RSI_SmoothedBuffer);
   if(copied<=0)
     {
      Print("Fallo al copiar el indicador suavizado RSI. Error = ",
            GetLastError(),",  copied = ",copied);
      return(0);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

Métodos de alisamiento

Muchos indicadores técnicos se basan sobre unos u otros métodos de alisamiento de las series de precios. Algunos indicadores técnicos estándares requieren la especificación del tipo de alisamiento como un parámetro de entrada. Los identificadores de la enumeración ENUM_MA_METHOD sirven para especificar el tipo deseado de alisamiento.

ENUM_MA_METHOD

Identificador	Descripción
MODE_SMA	Media móvil simple
MODE_EMA	Media móvil exponencial
MODE_SMMA	Media móvil suavizada
MODE_LWMA	Media móvil pronderada

Ejemplo:

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//---- handles for moving averages
int    ExtJawsHandle;
int    ExtTeethHandle;
int    ExtLipsHandle;
//--- get MA's handles
ExtJawsHandle=iMA(NULL,0,JawsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtTeethHandle=iMA(NULL,0,TeethPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtLipsHandle=iMA(NULL,0,LipsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
```

© 2000-2025, MetaQuotes Ltd.

Líneas de indicadores

Algunos indicadores técnicos tienen varios buffers dibujados en el gráfico. La numeración de los buffers de indicadores se empieza desde 0. A la hora de copiar los valores del indicador en un array del tipo double a través de la función CopyBuffer(), para algunos indicadores podemos indicar el identificador de un búfer copiado en vez de su número.

Identificadores de las líneas de indicadores admisibles a la hora de copiar los valores de indicadores iMACD(), iRVI() y iStochastic()

Constante	Valor	Descripción
MAIN_LINE	0	Línea principal
SIGNAL_LINE	1	Línea de señales

Identificadores de las líneas de indicadores admisibles a la hora de copiar los valores de indicadores ADX() y ADXW()

Constante	Valor	Descripción
MAIN_LINE	0	Línea principal
PLUSDI_LINE	1	Línea +DI
MINUSDI_LINE	2	Línea -DI

Identificadores de las líneas de indicadores admisibles a la hora de copiar los valores de indicador iBands()

Constante	Valor	Descripción
BASE_LINE	0	Línea principal
UPPER_BAND	1	Límite superior
LOWER_BAND	2	Límite inferior

Identificadores de las líneas de indicadores admisibles a la hora de copiar los valores de indicadores iEnvelopes() y iFractals()

Constante	Valor	Descripción
UPPER_LINE	0	Línea superior
LOWER_LINE	1	Línea inferior

Identificadores de las líneas de indicadores admisibles a la hora de copiar los valores de indicador iGator()

Constante	Valor	Descripción
UPPER_HISTOGRAM	0	Histograma de arriba
LOWER_HISTOGRAM	2	Histograma de abajo

© 2000-2025, MetaQuotes Ltd.

Identificadores de las líneas de indicadores admisibles a la hora de copiar los valores de indicador iAlligator()

Constante	Valor	Descripción
GATORJAW_LINE	0	Línea de mandíbulas
GATORTEETH_LINE	1	Línea de dientes
GATORLIPS_LINE	2	Línea de labios

Identificadores de las líneas de indicadores admisibles a la hora de copiar los valores de indicador iIchimoku()

Constante	Valor	Descripción
TENKANSEN_LINE	0	Línea Tenkan-sen
KIJUNSEN_LINE	1	Línea Kijun-sen
SENKOUSPANA_LINE	2	Línea Senkou Span A
SENKOUSPANB_LINE	3	Línea Senkou Span B
CHIKOUSPAN_LINE	4	Línea Chikou Span

© 2000-2025, MetaQuotes Ltd.

Estilos de dibujo

Cuando creamos un indicador personalizado se puede indicar uno de 18 tipos de construcción gráfica (modo de visualizar en la ventana principal o subventana del gráfico), cuyos valores se especifican en la enumeración ENUM_DRAW_TYPE.

En un indicador personalizado se permite usar cualquier tipo de construcción/dibujo de indicadores. Cada tipo de construcción requiere que se indique de uno a cinco arrays globales para almacenar los datos necesarios para dibujar. Hay que enlazar estas matrices de datos con los buffers de indicadores mediante la función SetIndexBuffer(), y especificar el tipo de datos de la enumeración ENUM_INDEXBUFFER_TYPE para cada buffer.

Dependiendo del estilo de dibujo, podemos necesitar de uno a cuatro buffers de valores (marcados como INDICATOR_DATA). Si un estilo admite la alternación dinámica de colores (todos los colores contienen la palabra COLOR en sus nombres), entonces necesitamos un buffer de color (tipo indicado INDICATOR_COLOR_INDEX). El buffer de colores siempre se enlaza después de los buffers de valores que corresponden al estilo.

ENUM_DRAW_TYPE

Identificador	Descripción	Buffer de valores	Buffer de color
DRAW_NONE	No se dibuja	1	0
DRAW_LINE	Línea	1	0
DRAW_SECTION	Sección	1	0
DRAW_HISTOGRAM	Histograma de la línea cero	1	0
DRAW_HISTOGRAM2	Histograma de dos buffers de indicadores	2	0
DRAW_ARROW	Dibujo de flechas	1	0
DRAW_ZIGZAG	Estilo Zigzag permite secciones verticale	2	0

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Buffer de valores	Buffer de color
	s en la barra		
DRAW_FILLING	Relleno de color entre dos niveles	2	0
DRAW_BARS	Visualización como secuencia de barras	4	0
DRAW_CANDLES	Visualización como secuencia de velas	4	0
DRAW_COLOR_LINE	Línea multicolor	1	1
DRAW_COLOR_SECTION	Secciones multicolor	1	1
DRAW_COLOR_HISTOGRAM	Histograma multicolor desde la línea cero	1	1
DRAW_COLOR_HISTOGRAM2	Histograma multicolor de dos buffers de indicadores	2	1
DRAW_COLOR_ARROW	Dibujo con	1	1

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Buffer de valores	Buffer de color
	flechas multicolor		
DRAW_COLOR_ZIGZAG	ZigZag multicolor	2	1
DRAW_COLOR_BARS	Barras multicolor	4	1
DRAW_COLOR_CANDLES	Velas multicolor	4	1

Para perfeccionar la visualización del tipo seleccionado de dibujo podemos usar los identificadores que vienen en la enumeración ENUM_PLOT_PROPERTY.

Para las funciones PlotIndexSetInteger() y PlotIndexGetInteger()

ENUM_PLOT_PROPERTY_INTEGER

Identificador	Descripción	Tipo de la propiedad
PLOT_ARROW	Código de flecha para el estilo DRAW_ARROW	uchar
PLOT_ARROW_SHIFT	Desplazamiento vertical de flechas para el estilo DRAW_ARROW	int
PLOT_DRAW_BEGIN	Número de barras iniciales sin dibujar y valores en DataWindow	int
PLOT_DRAW_TYPE	Tipo de construcción gráfica	ENUM_DRAW_TYPE
PLOT_SHOW_DATA	Indica la visualización de valores de construcción en la ventana DataWindow	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
PLOT_SHIFT	Desplazamiento de representación gráfica del indicador respecto al eje de tiempo en barras	int
PLOT_LINE_STYLE	Estilo de la línea de dibujo	ENUM_LINE_STYLE
PLOT_LINE_WIDTH	Grosor de línea de dibujo	int
PLOT_COLOR_INDEXES	Número de colores	int
PLOT_LINE_COLOR	Índice del buffer que contiene el color	color modificador=número del índice de color

Para la función PlotIndexSetDouble()

ENUM_PLOT_PROPERTY_DOUBLE

Identificador	Descripción	Tipo de la propiedad
PLOT_EMPTY_VALUE	Valor vacío para una representación gráfica, para la que no hay dibujo	double

Para la función PlotIndexSetString()

ENUM_PLOT_PROPERTY_STRING

Identificador	Descripción	Tipo de la propiedad
PLOT_LABEL	Nombre de la serie gráfica de indicador para la visualización en DataWindow. Para los estilos gráficos complejos, que requieren varios búferes de indicador para su	string

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	visualización, los nombres para cada búfer se puede establecer utilizando ";" como separador. El ejemplo del código se puede encontrar en DRAW_CANDLES	

Hay 5 estilos que se usan para dibujar una línea en un indicador personalizado. Se puede usarlas sólo si tienen el grosor de 0 a 1.

ENUM_LINE_STYLE

Identificador	Descripción
STYLE_SOLID	Línea continua
STYLE_DASH	Polilínea
STYLE_DOT	Línea de puntos
STYLE_DASHDOT	Línea de punto y raya
STYLE_DASHDOTDOT	Línea de dos puntos y raya

Para definir el estilo de dibujo de línea y el tipo de dibujo, se usa la función PlotIndexSetInteger(). A través de la función ObjectSetInteger() se puede determinar el espesor y el estilo del dibujo de niveles para las extensiones de Fibonacci.

Ejemplo:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- indicator buffers
double         MABuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- enlace del array con el búfer de indicador con el índice 0
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- establecer el dibujo de línea
   PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- configuración del estilo para dibujar la línea
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- color de línea
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
//--- espesor de lines
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
//--- etiqueta para la línea
   PlotIndexSetString(0,PLOT_LABEL,"Moving Average");
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   for(int i=prev_calculated;i<rates_total;i++)
     {
      MABuffer[i]=close[i];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

Propiedades de indicadores personalizados

El número de los buffers de indicadores que se puede usar en un indicador personalizado no está limitado. Pero para cada array que se designa como búfer de indicador a través de la función SetIndexBuffer() es necesario indicar el tipo de datos que él va a almacenar. Puede ser uno de los valores de la enumeración ENUM_INDEXBUFFER_TYPE.

ENUM_INDEXBUFFER_TYPE

Identificador	Descripción
INDICATOR_DATA	Datos para dibujar
INDICATOR_COLOR_INDEX	Colores
INDICATOR_CALCULATIONS	Buffers auxiliares para los cálculos intermedios

Un indicador personalizado cuenta con una variedad de configuraciones para proporcionar una visualización y percepción convenientes. Estas configuraciones se realizan a través de asignación de las propiedades de indicador correspondientes usando las funciones IndicatorSetDouble(), IndicatorSetInteger() y IndicatorSetString(). Los identificadores de las propiedades del indicador se encuentran en la enumeración ENUM_CUSTOMIND_PROPERTY.

ENUM_CUSTOMIND_PROPERTY_INTEGER

Identificador	Descripción	Tipo de la propiedad
INDICATOR_DIGITS	Precisión de dibujo de los valores del indicador	int
INDICATOR_HEIGHT	El alto fijo de la propia ven	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	tana del indicador (comando del preprocesador #property indicator_height)	
INDICATOR_LEVELS	Número de niveles en la ventana del indicador	int
INDICATOR_LEVELCOLOR	Color de la línea	color modificador - número de nivel

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	del nivel	
INDICATOR_LEVELSTYLE	Estilo de la línea del nivel	ENUM_LINE_STYLE modificador - número de nivel
INDICATOR_LEVELWIDTH	Grosor de la línea del nivel	int modificador - número de nivel
INDICATOR_FIXED_MINIMUM	Mínimo fijo para la ventana del indicador. La propiedad está disponible solo par	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	a la escritura por parte de la función IndicatorSetInteger()	
INDICATOR_FIXED_MAXIMUM	Máximo fijo para a ventana del indicador. La propiedad está disponible solo para la escritu	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	ra por parte de la función IndicatorSetInteger()	

ENUM_CUSTOMIND_PROPERTY_DOUBLE

Identificador	Descripción	Tipo de la propiedad
INDICATOR_MINIMUM	Mínimo de la ventana del indicador	double
INDICATOR_MAXIMUM	Máximo de la ventana del indicador	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad	
INDICATOR_LEVELVALUE	Valor del nivel	double	modificador - número de nivel

ENUM_CUSTOMIND_PROPERTY_STRING

Identificador	Descripción	Tipo de la propiedad	
INDICATOR_SHORTNAME	Nombre breve del indicador	string	
INDICATOR_LEVELTEXT	Descripción del nivel	string	modificador - número de nivel

Ejemplos:

```
//--- indicator settings
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_color2  clrRed
//--- input parameters
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- indicator buffers
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
```

© 2000-2025, MetaQuotes Ltd.

```
double LowesBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
   SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- set accuracy
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- set levels
   IndicatorSetInteger(INDICATOR_LEVELS,2);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- set maximum and minimum for subwindow
   IndicatorSetDouble(INDICATOR_MINIMUM,0);
   IndicatorSetDouble(INDICATOR_MAXIMUM,100);
//--- sets first bar from what index will be drawn
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
   PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- set style STYLE_DOT for second line
   PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- name for DataWindow and indicator subwindow label
   IndicatorSetString(INDICATOR_SHORTNAME,"Stoch("+KPeriod+","+DPeriod+","+Slowing+")"
   PlotIndexSetString(0,PLOT_LABEL,"Main");
   PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- sets drawing line to empty value
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- initialization done
  }
```

© 2000-2025, MetaQuotes Ltd.

Tipos de indicadores técnicos

Hay dos opciones de crear de forma programada un manejador del indicador para el acceso posterior a sus valores. El primer modo consiste en indicar directamente el nombre de la función de la lista de indicadores técnicos. El segundo modo permite, usando la función IndicatorCreate(), crear uniformemente un manejador de cualquier indicador asignando un identificador de la enumeración ENUM_INDICATOR. Ambas opciones valen para crear un manejador, podemos usar la opción que nos parezca más apropiada en cada caso particular durante el desarrollo de un programa en MQL5.

Cuando se crea un indicador del tipo IND_CUSTOM, el campo *type* del primer elemento de un array de parámetros de entrada MqlParam debe tener obligatoriamente el valor TYPE_STRING de la enumeración ENUM_DATATYPE, mientras que el campo *string_value* del primer elemento debe tener el nombre del indicador personalizado.

ENUM_INDICATOR

Identificador	Indicador
IND_AC	Accelerator Oscillator
IND_AD	Accumulation/Distribution
IND_ADX	Average Directional Index
IND_ADXW	ADX by Welles Wilder
IND_ALLIGATOR	Alligator
IND_AMA	Adaptive Moving Average
IND_AO	Awesome Oscillator
IND_ATR	Average True Range
IND_BANDS	Bollinger Bands®
IND_BEARS	Bears Power
IND_BULLS	Bulls Power
IND_BWMFI	Market Facilitation Index
IND_CCI	Commodity Channel Index
IND_CHAIKIN	Chaikin Oscillator
IND_CUSTOM	Custom indicator
IND_DEMA	Double Exponential Moving Average
IND_DEMARKER	DeMarker
IND_ENVELOPES	Envelopes
IND_FORCE	Force Index
IND_FRACTALS	Fractals
IND_FRAMA	Fractal Adaptive Moving Average

© 2000-2025, MetaQuotes Ltd.

Identificador	Indicador
IND_GATOR	Gator Oscillator
IND_ICHIMOKU	Ichimoku Kinko Hyo
IND_MA	Moving Average
IND_MACD	MACD
IND_MFI	Money Flow Index
IND_MOMENTUM	Momentum
IND_OBV	On Balance Volume
IND_OSMA	OsMA
IND_RSI	Relative Strength Index
IND_RVI	Relative Vigor Index
IND_SAR	Parabolic SAR
IND_STDDEV	Standard Deviation
IND_STOCHASTIC	Stochastic Oscillator
IND_TEMA	Triple Exponential Moving Average
IND_TRIX	Triple Exponential Moving Averages Oscillator
IND_VIDYA	Variable Index Dynamic Average
IND_VOLUMES	Volumes
IND_WPR	Williams' Percent Range

© 2000-2025, MetaQuotes Ltd.

Identificadores de tipos de datos

Cuando se crea un manejador del indicador a través de la función IndicatorCreate(), un array del tipo MqlParam tiene que ser especificado como el último parámetro. Como consecuencia, la estructura MqlParam, que describe los parámetros del indicador, contiene un campo especial *type*. Este campo contiene la información sobre el tipo de datos (tipo real, de números enteros o de cadena), los que se pasan a los elementos concretos de este array. El valor de este campo de la estructura MqlParam puede ser uno de los valores de la enumeración ENUM_DATATYPE.

ENUM_DATATYPE

Identificador	Tipo de datos
TYPE_BOOL	bool
TYPE_CHAR	char
TYPE_UCHAR	uchar
TYPE_SHORT	short
TYPE_USHORT	ushort
TYPE_COLOR	color
TYPE_INT	int
TYPE_UINT	uint
TYPE_DATETIME	datetime
TYPE_LONG	long
TYPE_ULONG	ulong
TYPE_FLOAT	float
TYPE_DOUBLE	double
TYPE_STRING	string

Cada elemento de este array describe un parámetro de entrada correspondiente de un indicador técnico que se crea, por eso el tipo y el orden de los elementos en el array tiene que ser mantenido estrictamente de acuerdo con la descripción.

© 2000-2025, MetaQuotes Ltd.

Estado de entorno

Las constantes que describen el entorno corriente de ejecución de un programa mql5 se dividen en dos grupos:

- Estado del terminal de cliente - información sobre el terminal de cliente;
- Información sobre el programa MQL5 en ejecución - propiedades de un programa mql5 que ayudan a dirigir adicionalmente su comportamiento;
- Información sobre el instrumento - obtención de información comercial sobre el instrumento;
- Información sobre la cuenta - información sobre la cuenta comercial corriente;
- Estadística de simulación - resultados de prueba del Asesor Experto.

© 2000-2025, MetaQuotes Ltd.

Estado del terminal de cliente

La información sobre el terminal de cliente se obtiene a través de estas dos funciones: TerminalInfoInteger() y TerminalInfoString(). Para parámetros, estas funciones aceptan valores de las enumeraciones ENUM_TERMINAL_INFO_INTEGER y ENUM_TERMINAL_INFO_STRING respectivamente.

ENUM_TERMINAL_INFO_INTEGER

Identificador	Descripción	Tipo de la propiedad
TERMINAL_BUILD	Número de la build del terminal	int
TERMINAL_COMMUNITY_ACCOUNT	Bandera de la presencia de los datos de autorización de MQL5.community en el terminal	bool
TERMINAL_COMMUNITY_CONNECTION	Conexión a MQL5.community	bool
TERMINAL_CONNECTED	Conexión al servidor comercial	bool
TERMINAL_DLLS_ALLOWED	Permiso de usar DLL	bool
TERMINAL_TRADE_ALLOWED	Permiso de hacer comercio	bool
TERMINAL_EMAIL_ENABLED	Permiso de enviar correo electrónico usando el servidor SMTP y nombre de usuario especificados en las configuraciones del terminal	bool
TERMINAL_FTP_ENABLED	Permiso de enviar informes a través de FTP a un servidor indicado para una cuenta comercial especificada en las configuraciones del terminal	bool
TERMINAL_NOTIFICATIONS_ENABLED	Permiso para enviar las notificaciones al smartphone	bool
TERMINAL_MAXBARS	Número máximo de barras en el gráfico	int
TERMINAL_MQID	Bandera de presencia de MetaQuotes ID para el envío de las notificaciones Push	int
TERMINAL_CODEPAGE	Número de la página de código del idioma instalado en el terminal de cliente	int
TERMINAL_CPU_CORES	Número de procesadores en el sistema	int
TERMINAL_DISK_SPACE	Tamaño de la memoria libre en el disco para la carpeta MQL5\Files	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	del terminal (agente), en MB	
TERMINAL_MEMORY_PHYSICAL	Tamaño de la memoria física en el sistema, en MB	int
TERMINAL_MEMORY_TOTAL	Tamaño de la memoria disponible para el proceso del terminal (agente), en MB	int
TERMINAL_MEMORY_AVAILABLE	Tamaño de la memoria libre del proceso del terminal (agente), en MB	int
TERMINAL_MEMORY_USED	Tamaño de la memoria usada por el terminal (agente), en MB	int
TERMINAL_X64	Indicación "terminal de 64 bits"	bool
TERMINAL_OPENCL_SUPPORT	Versión de OpenCL soportada en el formato 0x00010002 = 1.2. "0" significa que OpenCL no se soporta	int
TERMINAL_SCREEN_DPI	La capacidad de resolución a la hora de mostrar información en la pantalla se mide por la cantidad de puntos por pulgada lineal de la superficie (DPI). El conocimiento de este parámetro permite definir las dimensiones de los objetos gráficos de tal forma que parezcan iguales en monitores con diferente capacidad de resolución.	int
TERMINAL_SCREEN_LEFT	Coordenada izquierda de la pantalla virtual. La pantalla virtual es un rectángulo que abarca todo el monitor. Si en el sistema hay dos monitores y su orden se ha establecido de derecha a izquierda, la coordenada izquierda de la pantalla virtual puede encontrarse en el límite de los dos monitores.	int
TERMINAL_SCREEN_TOP	Coordenada superior de la pantalla virtual	int
TERMINAL_SCREEN_WIDTH	Anchura del terminal	int
TERMINAL_SCREEN_HEIGHT	Altura del terminal	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
TERMINAL_LEFT	Coordenada izquierda del terminal con respecto a la pantalla virtual	int
TERMINAL_TOP	Coordenada superior del terminal con respecto a la pantalla virtual	int
TERMINAL_RIGHT	Coordenada derecha del terminal con respecto a la pantalla virtual	int
TERMINAL_BOTTOM	Coordenada inferior del terminal con respecto a la pantalla virtual	int
TERMINAL_PING_LAST	Último valor conocido del ping hasta el servidor comercial en microsegundos. En un segundo hay un millón de microsegundos.	int
TERMINAL_VPS	Señal de que el terminal está siendo ejecutado en el servidor virtual MetaTrader Vitrtual Hosting (MetaTrader VPS)	bool
Identificador de la tecla	**Descripción**	
TERMINAL_KEYSTATE_LEFT	Estado de la tecla "Flecha izquierda"	int
TERMINAL_KEYSTATE_UP	Estado de la tecla "Flecha arriba"	int
TERMINAL_KEYSTATE_RIGHT	Estado de la tecla "Flecha derecha"	int
TERMINAL_KEYSTATE_DOWN	Estado de la tecla "Flecha abajo"	int
TERMINAL_KEYSTATE_SHIFT	Estado de la tecla "Shift"	int
TERMINAL_KEYSTATE_CONTROL	Estado de la tecla "Ctrl"	int
TERMINAL_KEYSTATE_MENU	Estado de la tecla "Windows"	int
TERMINAL_KEYSTATE_CAPSLOCK	Estado de la tecla "CapsLock"	int
TERMINAL_KEYSTATE_NUMLOCK	Estado de la tecla "NumLock"	int
TERMINAL_KEYSTATE_SCRLOCK	Estado de la tecla "ScrollLock"	int
TERMINAL_KEYSTATE_ENTER	Estado de la tecla "Enter"	int
TERMINAL_KEYSTATE_INSERT	Estado de la tecla "Insert"	int
TERMINAL_KEYSTATE_DELETE	Estado de la tecla "Delete"	int
TERMINAL_KEYSTATE_HOME	Estado de la tecla "Home"	int
TERMINAL_KEYSTATE_END	Estado de la tecla "End"	int
TERMINAL_KEYSTATE_TAB	Estado de la tecla "Tab"	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
TERMINAL_KEYSTATE_PAGEUP	Estado de la tecla "PageUp"	int
TERMINAL_KEYSTATE_PAGEDOWN	Estado de la tecla "PageDown"	int
TERMINAL_KEYSTATE_ESCAPE	Estado de la tecla "Escape"	int

La llamada de TerminalInfoInteger(TERMINAL_KEYSTATE_XXX) retorna el mismo código de estado de una tecla que la función GetKeyState() de MSDN.

Ejemplo de cálculo del coeficiente de escala:

```
//--- creamos un botyn con una anchura de 1.5 pulgadas en la pantalla
int screen_dpi = TerminalInfoInteger(TERMINAL_SCREEN_DPI); // obtenemos los puntos por
int base_width = 144;                                      // anchura bősica en puntos
int width      = (button_width * screen_dpi) / 96;         // calculamos la anchura de
...

//--- cőlculo del coeficiente de escala en tanto por ciento
int scale_factor=(TerminalInfoInteger(TERMINAL_SCREEN_DPI) * 100) / 96;
//--- uso del coeficiente de escala
width=(base_width * scale_factor) / 100;
```

Usando el recurso gráfico de esta forma, tendrá el mismo tamaño a simple vista en monitores con diferentes capacidades de resolución. Además, las dimensiones de los elementos de control (botones, ventanas de diálogo, etc) se corresponderán con los ajustes de la personalización.

ENUM_TERMINAL_INFO_DOUBLE

Identificador	Descripción	Tipo de la propiedad
TERMINAL_COMMUNITY_BALANCE	Balance del usuario en MQL5.community	double
	Porcentaje de paquetes de red nuevamente enviados en protocolo TCP/IP para todos los servicios y aplicaciones iniciados en la computadora dada. Incluso en la red más rápida y mejor configurada tienen lugar pérdidas de paquetes y, como consecuencia, no se da la confirmación de la entrega de los paquetes entre el receptor y el emisor. En tales casos, el paquete "perdido" es enviado de nuevo.	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	No es un índice de calidad de la conexión de un terminal concreto a un servidor comercial concreto, puesto que se calcula para toda la actividad de la red, incluyendo la actividad sistémica y la actividad de fondo. El índice TERMINAL_RETRANSMISSION se solicita una vez por minuto desde el sistema operativo. El propio terminal no calcula este índice.	

Las operaciones de archivos pueden realizarse sólo en dos directorios; las rutas correspondientes se puede obtener llamando a las propiedades de TERMINAL_DATA_PATH y TERMINAL_COMMONDATA_PATH.

ENUM_TERMINAL_INFO_STRING

Identificador	Descripción	Tipo de la propiedad
TERMINAL_LANGUAGE	Idioma del terminal	string
TERMINAL_COMPANY	Nombre de la empresa	string
TERMINAL_NAME	Nombre del terminal	string
TERMINAL_PATH	Carpeta de la que se inicia el terminal	string
TERMINAL_DATA_PATH	Carpeta donde se almacenan los datos del terminal	string
TERMINAL_COMMONDATA_PATH	Carpeta general de todos los terminales de cliente instalados en el ordenador	string
TERMINAL_CPU_NAME	Nombre del procesador	string
TERMINAL_CPU_ARCHITECTURE	Arquitectura del procesador	string
TERMINAL_OS_VERSION	Nombre del sistema operativo del usuario	string
TERMINAL_COLORTHEME_NAME	Esquema de color del terminal: Light y Dark.	string

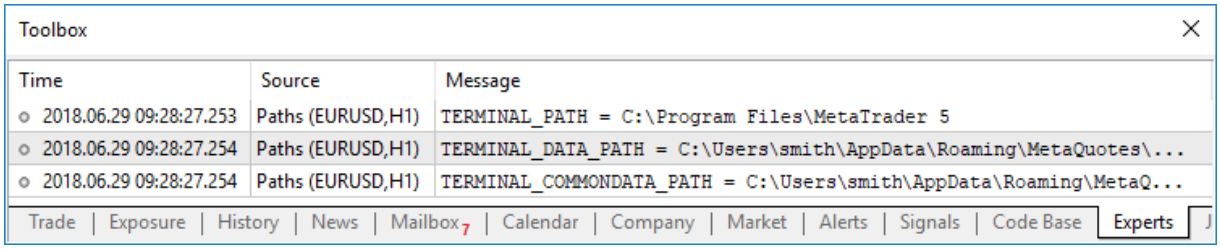
© 2000-2025, MetaQuotes Ltd.

Para mejor entendimiento de las rutas que se guardan en las propiedades de los parámetros TERMINAL_PATH, TERMINAL_DATA_PATH y TERMINAL_COMMONDATA_PATH, se recomienda ejecutar el script que devolverá estos valores para la copia del terminal instalada en su ordenador.

Ejemplo: Script devuelve la información sobre las rutas del terminal de cliente

```
//+------------------------------------------------------------------+
//|                                          Check_TerminalPaths.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
   Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
   Print("TERMINAL_COMMONDATA_PATH = ",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
  }
```

Como resultado de su ejecución, en el Registro de Asesores Expertos aparecerán los mensajes parecidos a los que vienen más abajo.



Obtener información sobre el esquema de color del terminal

El terminal le permite establecer esquemas de color: claro (por defecto) y oscuro. Al crear aplicaciones de usuario que tienen una interfaz gráfica para interactuar con el usuario, los desarrolladores deberán considerar el esquema de color actual del terminal. Por lo tanto, para mayor comodidad de los usuarios, se hace necesaria una adaptación dinámica de los componentes visuales utilizados en la aplicación.

Con este propósito, se han agregado funciones para determinar el esquema de color del terminal:

- El valor TERMINAL_COLORTHEME_NAME de la enumeración ENUM_TERMINAL_INFO_STRING permite obtener el nombre del esquema de color con la ayuda de TerminalInfoString. Valores posibles: Light y Dark.

© 2000-2025, MetaQuotes Ltd.

- Use las propiedades THEME_COLOR_* de la enumeración ENUM_TERMINAL_INFO_INTEGER para obtener colores de elementos específicos de la interfaz con la ayuda de TerminalInfoInteger.

Identificador	Descripción	Tipo de propiedad
THEME_COLOR_WINDOW	Fondo de la ventana	color
THEME_COLOR_WINDOWTEXT	Texto en la ventana	color
THEME_COLOR_WINDOW	Fondo de la ventana	color
THEME_COLOR_WINDOWTEXT	Texto en la ventana	color
THEME_COLOR_BTNTEXT	Texto en los botones	color
THEME_COLOR_GRAYTEXT	Texto inactivo (desactivado)	color
THEME_COLOR_INFOTEXT	Texto en las pistas	color
THEME_COLOR_INFOBK	Fondo de las pistas	color
THEME_COLOR_3DFACE	Parte frontal de los elementos 3D	color
THEME_COLOR_3DLIGHT	Parte clara de los elementos 3D	color
THEME_COLOR_3DSHADOW	Parte oscura de los elementos 3D	color
THEME_COLOR_3DDKSHADOW	Sombra oscura de elementos 3D	color
THEME_COLOR_3DHILIGHT	Iluminación de elementos 3D	color
THEME_COLOR_HIGHLIGHT	Fondo de los elementos seleccionados	color
THEME_COLOR_HIGHLIGHTTEXT	Texto de los elementos seleccionados	color
THEME_COLOR_BTNFACE	Parte frontal de los botones	color
THEME_COLOR_BTNHILIGHT	Iluiminación de los botones	color
THEME_COLOR_BTNSHADOW	Sombra de los botones	color
THEME_COLOR_MENU	Fondo del menú	color
THEME_COLOR_MENUBAR	Fondo del panel del menú	color
THEME_COLOR_MENUTEXT	Texto del menú	color
THEME_COLOR_MENUHILIGHT	Iluminación del punto seleccionado del menú	color
THEME_COLOR_ACTIVECAPTION	Encabezado de la ventana activa	color
THEME_COLOR_INACTIVECAPTION	Encabezado de la ventana inactiva	color

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de propiedad
THEME_COLOR_GRADIENTINACTIVECAPTION	Gradiente del encabezado de la ventana inactiva	color
THEME_COLOR_CAPTIONTEXT	Texto del encabezado de la ventana	color
THEME_COLOR_INACTIVECAPTIONTEXT	Texto del encabezado de la ventana inactiva	color
THEME_COLOR_HOTTEXT	Hipervínculos o elementos activos	color
THEME_COLOR_NONE	Color no seleccionado	color
THEME_COLOR_SEPARATOR	Separador	color
THEME_COLOR_SCROLLBACK	Barra de desplazamiento	color
THEME_COLOR_LINE1	Color de fondo de una línea impar en el Diario	color
THEME_COLOR_LINE2	Color de fondo de una línea par en el Diario	color
THEME_COLOR_GRID	Color de la cuadrícula en el Diario	color
THEME_COLOR_SUMMARY	Color de fondo de la línea final en el Diario	color
THEME_COLOR_ERROR	Color del texto sobre errores	color
THEME_COLOR_INVALID	Color del texto sobre valor no válido	color
THEME_COLOR_NEGATIVE	Color del valor negativo	color
THEME_COLOR_POSITIVE	Color del valor positivo	color
THEME_COLOR_LINK	Color del enlace	color
THEME_COLOR_LINKHOVER	Color del enlace al colocar el cursor	color
THEME_COLOR_LINKTESTER	Color de los enlaces en la caché de resultados de los anteriores inicios de simulación/optimización	color
THEME_COLOR_TEXTUP	Botón no pulsado	color
THEME_COLOR_TEXTDOWN	Botón pulsado	color
THEME_COLOR_BACKUP	Color de los botones "BUY" y "SELL" en la ventana "One Click	color

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de propiedad
	Trading" al aumentar el precio propuesto	
THEME_COLOR_BACKDOWN	Color de los botones "BUY" y "SELL" en la ventana "One Click Trading" al disminuir el precio propuesto	color
THEME_COLOR_CLOSE	Botón de la operación "Close"	color
THEME_COLOR_BUY	Botón de la operación "BUY"	color
THEME_COLOR_SELL	Botón de la operación "SELL"	color
THEME_COLOR_DEPOSIT	Color del botón "Deposit"	color
THEME_COLOR_WITHDRAWAL	Color del botón "Withdrawal"	color
THEME_COLOR_BID	Color de la línea "Bid"	color
THEME_COLOR_ASK	Color de la línea "Ask"	color
THEME_COLOR_STOPS	Color de la línea "Stop"	color
THEME_COLOR_STOPS_RED	Iluminación del valor StopLoss al darse en un valor negativo de beneficio en la pestaña Trading	color
THEME_COLOR_STOPS_GREEN	Iluminación del valor StopLoss al darse en un valor positivo de beneficio en la pestaña Trading	color
THEME_COLOR_CONFIRM	Color del botón "Accept" en la ventana de envío de órdenes	color
THEME_COLOR_REQUOTE	Color del botón "Requote" en la ventana de envío de órdenes	color
THEME_COLOR_REJECT	Color del botón "Reject" en la ventana de envío de órdenes	color
THEME_COLOR_NOTIFICATION	Color del texto sobre los cambios del servidor en la ventana de envío de órdenes	color
THEME_COLOR_RATING	Color de la franja de clasificación en el sistema de formación	color
THEME_COLOR_BOOK_BUY	Color de fondo de los niveles de compra en la Profundidad de mercado	color
THEME_COLOR_BOOK_SELL	Color de fondo de los niveles de venta en la Profundidad de	color

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de propiedad
	mercado	
THEME_COLOR_BOOK_LAST	Color de la última transacción en la Profundidad de mercado	color
THEME_COLOR_BOOK_STOP	Color del nivel de StopLoss en la Profundidad de mercado	color
THEME_COLOR_BOOK_SPREAD	Color de fondo de los niveles de spread en la Profundidad de mercado	color
THEME_COLOR_TICKS_BID	Color de la línea Bid del gráfico de ticks en la ventana de envío de órdenes	color
THEME_COLOR_TICKS_ASK	Color de la línea Ask del gráfico de ticks en la ventana de envío de órdenes	color
THEME_COLOR_TICKS_LAST	Color de la línea Last del gráfico de ticks en la ventana de envío de órdenes	color
THEME_COLOR_TICKS_CROSS	Color de la retícula en el gráfico de ticks en la ventana de envío de órdenes	color
THEME_COLOR_TICKS_SL	Color de la línea StopLoss en el gráfico de ticks en la ventana de envío de órdenes	color
THEME_COLOR_TICKS_TP	Color de la línea TakeProfit en el gráfico de ticks en la ventana de envío de órdenes	color
THEME_COLOR_TESTER_START	Color del botón "Start" de la simulación/optimización	color
THEME_COLOR_TESTER_STOP	Color del botón "Stop" de la simulación/optimización	color
THEME_COLOR_TESTER_START_FRAME	Color del marco del botón "Start" de la simulación/optimización	color
THEME_COLOR_TESTER_STOP_FRAME	Color del marco del botón "Stop" de la simulación/optimización	color
THEME_COLOR_TESTER_PROGRESS	Color de la barra de progreso de la simulación/optimización	color
THEME_COLOR_TESTER_BALANCE	Color de la línea de Balance en el simulador de estrategias	color

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de propiedad
THEME_COLOR_TESTER_EQUITY	Color de la línea de Equidad en el simulador de estrategias	color
THEME_COLOR_TESTER_MARGIN	Color del gráfico de carga del depósito en el simulador de estrategias	color
THEME_COLOR_PROFILER_CALL	Color de la línea de código con la llamada al realizar el Perfilado	color
THEME_COLOR_PROFILER_CALLSEL	Color de la línea de código destacada con la llamada al realizar el Perfilado	color
THEME_COLOR_PROFILER_LINE	Color de la línea en el Diario de Perfilado	color
THEME_COLOR_PROFILER_LINESEL	Color de la línea destacada en el Diario de Perfilado	color

© 2000-2025, MetaQuotes Ltd.

Información sobre el programa MQL5 en ejecución

Para obtener la información sobre los programas mql5 que actualmente están funcionando, se usan las constantes de las enumeraciones ENUM_MQL_INFO_INTEGER y ENUM_MQL_INFO_STRING.

Para la función MQLInfoInteger()

ENUM_MQL_INFO_INTEGER

Identificador	Descripción	Tipo de la propiedad
MQL_HANDLES_USED	Número actual de identificadores de objetos activos. Se consideran tanto objetos dinámicos (creados a través de new), así como objetos no dinámicos, variables globales/locales o miembros de clase. Cuantos más identificadores utilice un programa, más recursos consumirá.	int
MQL_MEMORY_LIMIT	Tamaño máximo posible de la memoria dinámica para el programa MQL5 en Mb	int
MQL_MEMORY_USED	Tamaño de la memoria usada por el programa MQL5 en Mb	int
MQL_PROGRAM_TYPE	Tipo del programa mql5	ENUM_PROGRAM_TYPE
MQL_DLLS_ALLOWED	Permiso de usar DLL para **este programa iniciado**	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
MQL_TRADE_ALLOWED	Permiso para tradear concedido a **este programa iniciado**	bool
MQL_SIGNALS_ALLOWED	Permiso para trabajar con las señales **de este programa iniciado**	bool
MQL_DEBUG	Indica que el programa iniciado trabaja en el modo de depuración	bool
MQL_PROFILER	Indica que el programa iniciado trabaja en el modo de perfiladura del código	bool
MQL_TESTER	Indica que el programa iniciado trabaja en el Probador	bool
MQL_FORWARD	Indica que el programa iniciado trabaja en el proceso de simulación en tiempo real (forward testing)	bool
MQL_OPTIMIZATION	Indica que el programa iniciado trabaja en el proceso de optimización	bool
MQL_VISUAL_MODE	Indica que el programa iniciado trabaja en el modo visual de simulación	bool
MQL_FRAME_MODE	Indica que el EA iniciado en el	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	gráfico trabaja en el modo de recogida de los frames de resultados de la optimización	
MQL_LICENSE_TYPE	Tipo de licencia del módulo EX5. La licencia se refiere al módulo EX5 desde el cual se hace la solicitud con el uso de MQLInfoInteger(MQL_LICENSE_TYPE).	ENUM_LICENSE_TYPE
MQL_STARTED_FROM_CONFIG	Retorna true si el script/asesor se inicia desde la sección StartUp del archivo de configuración. Esto significa que este script/asesor ha sido especificado en el archivo de configuración con el que ha sido iniciado el terminal.	bool

Para la función MQLInfoString()

ENUM_MQL_INFO_STRING

Identificador	Descripción	Tipo de la propiedad
MQL_PROGRAM_NAME	Nombre del programa mql5 en ejecución	string
MQL_PROGRAM_PATH	Ruta para **dicho programa en ejecución**	string

© 2000-2025, MetaQuotes Ltd.

Para obtener la información sobre el tipo del programa en ejecución podemos usar los valores de la enumeración ENUM_PROGRAM_TYPE.

ENUM_PROGRAM_TYPE

Identificador	Descripción
PROGRAM_SCRIPT	Script
PROGRAM_EXPERT	Experto
PROGRAM_INDICATOR	Indicador
PROGRAM_SERVICE	Servicio

ENUM_LICENSE_TYPE

Identificador	Descripción
LICENSE_FREE	Versión gratuita ilimitada
LICENSE_DEMO	Versión demo del producto de pago de la Tienda. Funciona sólo en el Probador de Estrategias
LICENSE_FULL	Esta es una versión con licencia adquirida, que permite al menos 5 activaciones. El vendedor podrá aumentar el número permitido de activaciones.
LICENSE_TIME	Esta es una versión con un término de licencia limitado

Ejemplo:

```
ENUM_PROGRAM_TYPE mql_program=(ENUM_PROGRAM_TYPE)MQLInfoInteger(MQL_PROGRAM_TYPE);
switch(mql_program)
  {
   case PROGRAM_SCRIPT:
     {
      Print(__FILE__+" is script");
      break;
     }
   case PROGRAM_EXPERT:
     {
      Print(__FILE__+" is Expert Advisor");
      break;
     }
   case PROGRAM_INDICATOR:
     {
      Print(__FILE__+" is custom indicator");
      break;
     }
   default:Print("MQL5 type value is ",mql_program);
```

© 2000-2025, MetaQuotes Ltd.

```
   //---
   Print("MQLInfoInteger(MQL_MEMORY_LIMIT)=", MQLInfoInteger(MQL_MEMORY_LIMIT), " MB")
   Print("MQLInfoInteger(MQL_MEMORY_USED)=", MQLInfoInteger(MQL_MEMORY_USED), " MB");
   Print("MQLInfoInteger(MQL_HANDLES_USED)=", MQLInfoInteger(MQL_HANDLES_USED), " hanc
```

© 2000-2025, MetaQuotes Ltd.

Información sobre el instrumento

Las funciones SymbolInfoInteger(), SymbolInfoDouble() y SymbolInfoString() sirven para obtener la información actual del mercado. Como segundo parámetro de estas funciones podemos pasar uno de los identificadores de las enumeraciones ENUM_SYMBOL_INFO_INTEGER, ENUM_SYMBOL_INFO_DOUBLE y ENUM_SYMBOL_INFO_STRING respectivamente.

Para la función SymbolInfoInteger()

ENUM_SYMBOL_INFO_INTEGER

Identificador	Descripción	Tipo de la propiedad
SYMBOL_SUBSCRIPTION_DELAY	Indica que los datos del símbolo van con retraso. La propiedad se puede solicitar solo para los símbolos seleccionado	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	sen MarketWatch (SYMBOL_SELECT = true). Para los símbolos no seleccionados, se generará el errorERR_MARKET_NOT_SELECTED (4302)	
SYMBOL_SECTOR	Sector	ENUM_SYMBOL_SECTOR

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	económico al que pertenece el símbolo.	
SYMBOL_INDUSTRY	Tipo de industria o sector económico al que pertenece el símbolo	ENUM_SYMBOL_INDUSTRY
SYMBOL_CUSTOM	Indica que el símbolo es personalizado, es	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	decir, es creado artificialmente con base en otros símbolos de la ventana MarketWatch o/y fuentes de datos externas	
SYMBOL_BACKGROUND_COLOR	Color del fondo con el que se	color

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	ilumina el símbolo en MarketWatch	
SYMBOL_CHART_MODE	Tipo de precio para la construcción de barras - Bid o Last	ENUM_SYMBOL_CHART_MODE
SYMBOL_EXIST	El símbolo con este nombre existe	bool
SYMBOL_SELECT	Indica que	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	el símbolo ha sido seleccionado en MarketWatch	
SYMBOL_VISIBLE	Indica que el símbolo elegido se representa en MarketWatch. Algunos símbolos (norm alm	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	ente se trata de cursos cruzados, necesarios para el cálculo de los requisitos del margen y el beneficio en la divisa del depósito) se selecciona de forma	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	automática pero, en este caso, podría no ser representados en MarketWatch. Para su representación, dichos símbolos deberán ser seleccion	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	ados de forma explícita.	
SYMBOL_SESSION_DEALS	Número de transacciones en la sesión actual	long
SYMBOL_SESSION_BUY_ORDERS	Número total de órdenes de compra en el momento actual	long
SYMBOL_SESSION_SELL_ORDERS	Núme	long

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	rototal de órdenes de venta en el momento actual	
SYMBOL_VOLUME	Volume - volumen en la última transacción	long
SYMBOL_VOLUMEHIGH	Volumen máximo del día	long
SYMBOL_VOLUMELOW	Volumen mínimo	long

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	del día	
SYMBOL_TIME	Hora de la última cotización	datetime
SYMBOL_TIME_MSC	Hora de la última cotización en milisegundos desde el 1970.01.01	long
SYMBOL_DIGITS	Número de dígitos después del punto dec	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	imal	
SYMBOL_SPREAD	Tamaño de spread en puntos	int
SYMBOL_TICKS_BOOKDEPTH	Cantidad máxima de las solicitudes mostradas en la profundidad. Para los instrumentos sin cola de solicitu	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	des, el valores 0	
SYMBOL_TRADE_CALC_MODE	Modo de calcular el coste del contrato	ENUM_SYMBOL_CALC_MODE
SYMBOL_TRADE_MODE	Tipo de ejecución de órdenes	ENUM_SYMBOL_TRADE_MODE
SYMBOL_START_TIME	Fecha de inicio de licitaciones por instrumento (norm	datetime

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	almente se utiliza para los futuros)	
SYMBOL_EXPIRATION_TIME	Fecha final de licitaciones por herramienta (normalmente se utiliza para los futuros)	datetime
SYMBOL_TRADE_STOPS_LEVEL	Margen mínimo en pun	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	tos del actual precio de cierre para la colocación de las órdenes Stop	
SYMBOL_TRADE_FREEZE_LEVEL	Distancia de congelamiento de operaciones comerciales (en puntos)	int
SYMBOL_TRADE_EXEMODE	Modo	ENUM_SYMBOL_TRADE_EXECUTION

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	de ejecución de transacciones	
SYMBOL_SWAP_MODE	Modelo para calcular swap	ENUM_SYMBOL_SWAP_MODE
SYMBOL_SWAP_ROLLOVER3DAYS	Día de la semana para cargar la refinanciación del swap de 3 días	ENUM_DAY_OF_WEEK
SYMBOL_MARGIN_HEDGED_USE_LEG	Modo de	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	cálculo del margen cubierto por el lado mayor (Buy o Sell)	
SYMBOL_EXPIRATION_MODE	Banderas de los modos de expiración de la orden permitidos	int
SYMBOL_FILLING_MODE	Banderas de los modos de	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	relleno de la orden permitidos	
SYMBOL_ORDER_MODE	Banderas de los tipos de la orden permitidos	int
SYMBOL_ORDER_GTC_MODE	Plazo de expiración de las órdenes StopLoss y TakeProfit, si SYMBOL	ENUM_SYMBOL_ORDER_GTC_MODE

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	_EXPIRATION_MODE=SYMBOL_EXPIRATION_GTC (Good till cancelled)	
SYMBOL_OPTION_MODE	Tipo de opción	ENUM_SYMBOL_OPTION_MODE
SYMBOL_OPTION_RIGHT	Derecho de la opción (Call/Put)	ENUM_SYMBOL_OPTION_RIGHT

Para la función SymbolInfoDouble()

ENUM_SYMBOL_INFO_DOUBLE

Identificador	Descripción	Tipo de la propiedad
SYMBOL_BID	Bid - mejor	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	oferta de venta	
SYMBOL_BIDHIGH	Bid máximo del día	double
SYMBOL_BIDLOW	Bid mínimo del día	double
SYMBOL_ASK	Ask - mejor oferta de compra	double
SYMBOL_ASKHIGH	Ask máximo del día	double
SYMBOL_ASKLOW	Ask mínimo del día	double
SYMBOL_LAST	Precio de la última transacción	double
SYMBOL_LASTHIGH	Last máximo del día	double
SYMBOL_LASTLOW	Last mínimo del día	double
SYMBOL_VOLUME_REAL	Volume - volumen en la última transacción	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
SYMBOL_VOLUMEHIGH_REAL	Volumen máximo del día	double
SYMBOL_VOLUMELOW_REAL	Volumen mínimo del día	double
SYMBOL_OPTION_STRIKE	Precio de ejecución de la opción. Es el precio por el que el comprador de la opción puede comprar (si la opción es Call) o vender (si la opción es Put) un activo base, y el vendedor de la opción, consecuentemente, está obligado a vender	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	o comprar la cantidad correspondiente del activo base	
SYMBOL_POINT	Valor de un punto	double
SYMBOL_TRADE_TICK_VALUE	Valor SYMBOL_TRADE_TICK_VALUE_PROFIT	double
SYMBOL_TRADE_TICK_VALUE_PROFIT	Precio calculado del tick para la posición rentable	double
SYMBOL_TRADE_TICK_VALUE_LOSS	Precio calculado del tick para la posición no rentable	double
SYMBOL_TRADE_TICK_SIZE	Mínimo cambio de precio	double
SYMBOL_TRADE_CONTRACT_SIZE	Tamaño del contrat	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	o comercial	
SYMBOL_TRADE_ACCRUED_INTEREST	Ingreso de cupón acumuladoparte del ingreso de cupón de interes es para la obligación que se calcula proporcionalmente al número de días transcurridos desde la fecha de emisión de la obligación de cupón o la fecha del pago del ingreso de cupón anterior	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
SYMBOL_TRADE_FACE_VALUE	Valor facial - precio inicial de la obligación, establecido por el emisor	double
SYMBOL_TRADE_LIQUIDITY_RATE	Coeficiente de liquidez - parte del coste del activo que se puede usar como margen.	double
SYMBOL_VOLUME_MIN	Volumen mínimo para celebrar una transacción	double
SYMBOL_VOLUME_MAX	Volumen máximo para celebrar una transacción	double
SYMBOL_VOLUME_STEP	Paso mínimo del cambio	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	de volumen para celebrar una transacción	
SYMBOL_VOLUME_LIMIT	Volumen total máximo permitido de la posición abierta y órdenes pendientes (independientemente de la dirección) para un símbolo	double
SYMBOL_SWAP_LONG	Valor de swap long	double
SYMBOL_SWAP_SHORT	Valor de swap short	double
SYMBOL_SWAP_SUNDAY	Coeficiente de cálculo de swaps (SYMBOL_SW	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	AP_LONG o SYMBOL_SWAP_SHORT) al trasladar una posición desde el día indicado (SUNDAY) al siguiente. Pueden darse los siguientes valores: • 0 - los swaps no se calculan	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	• 1 - cálculo único swaps • 3 - cálculo triple swaps	
SYMBOL_SWAP_MONDAY	Coeficiente de cálculo de	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	swaps (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) al trasladar una posición desde el lunes al martes	
SYMBOL_SWAP_TUESDAY	Coeficiente de cálculo de swaps (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) al trasladar una posición desde el martes al miércoles	double
SYMBOL_SWAP_WEDNESDAY	Coeficiente de cálculo de swaps (SYMBOL_SW	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	AP_LONG o SYMBOL_SWAP_SHORT) al trasladar una posición desde el miércoles al jueves	
SYMBOL_SWAP_THURSDAY	Coeficiente de cálculo de swaps (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) al trasladar una posición desde el jueves al viernes	double
SYMBOL_SWAP_FRIDAY	Coeficiente de cálculo de swaps (SYMBOL_SWAP_LONG o SYMBO	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	L_SWAP_SHORT) al trasladar una posición desde el viernes al sábado	
SYMBOL_SWAP_SATURDAY	Coeficiente de cálculo de swaps (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) al trasladar una posición desde el sábado al domingo	double
SYMBOL_MARGIN_INITIAL	Margen inicial (inicializador) significa el monto de fondos asegurados necesa	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	rios en la moneda de margen para abrir posiciones en el volumen de un lote. Se utiliza para verificar los fondos del cliente al entrar en el merado. La función SymbolInfoMarginRate() ofrece información sobre la cantidad del margen cobrado dependiendo del tipo y la direcci	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	ón de la orden.	
SYMBOL_MARGIN_MAINTENANCE	Margen de mantenimiento del instrumento. En caso de establecerlo - indica el monto del margen en la moneda de margen del instrumento que se retiene de un lote. Se utiliza para verificar los fondos del cliente cuando se cambia el estado de la cuenta del cliente. Si el	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	margen de mantenimiento es igual a 0, se utiliza el margen inicial. La función SymbolInfoMarginRate() ofrece información sobre la cantidad del margen cobrado dependiendo del tipo y la dirección de la orden.	
SYMBOL_SESSION_VOLUME	Volumen total de transacciones de la sesión actual	double
SYMBOL_SESSION_TURNOVER	Circulación total	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	durante la sesión actual	
SYMBOL_SESSION_INTEREST	Volumen total de posiciones abiertas	double
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Volumen total de órdenes de compra en el momento actual	double
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Volumen total de órdenes de venta en el momento actual	double
SYMBOL_SESSION_OPEN	Precio de apertura de la sesión	double
SYMBOL_SESSION_CLOSE	Precio de cierre de la sesión	double
SYMBOL_SESSION_AW	Precio medio ponderado de	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	la sesión	
SYMBOL_SESSION_PRICE_SETTLEMENT	Precio de entrega para la sesión actual	double
SYMBOL_SESSION_PRICE_LIMIT_MIN	Precio mínimo aceptable para la sesión	double
SYMBOL_SESSION_PRICE_LIMIT_MAX	Precio máximo aceptable para la sesión	double
SYMBOL_MARGIN_HEDGED	Tamaño del contrato o el margen para un lote de posiciones solapadas (dos posiciones de un mismo símbolo dirigidas en dirección opuesta). Existen	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	dos métodos para calcular el margen de las posiciones solapadas. El método de cálculo lo determina el bróker. Cálculo básico: • Si para el instrumento se ha establecido un margen inicial (SYMBOL_MARGIN_INITIAL), entonces el margen cubie	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	rto se indica como valor absoluto (en dinero). • Si no se ha establecido un margen inicial (igual a 0), entonces en SYMBOL_MARGIN_HEDGED se indica el tamaño del contrato, que se usará al calcular el marg	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	en según la fórmula correspondiente al tipo de instrumento comercial (SYMBOL_TRADE_CALC_MODE). Cálculo según la posición mayor • El valor SYMBOL_MARGIN_HEDGED no se tiene en cuenta. • Se calcula el volumen	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	de todas las posiciones largas y cortas del instrumento. • Para cada lado se calcula el precio promedio ponderado de apertura, así como el precio promedio ponderado de conversión a la divisa del depósito. • Siguiendo con	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	las fórmulas correspondiente al tipo de instrumento (SYMBOL_TRADE_CALC_MODE), se calcula el margen para el lado corto y el largo. • Como valor final se usa el más alto.	
SYMBOL_PRICE_CHANGE	Medición del precio actual con respecto al final	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	del anterior día comercial, medida en tanto por ciento	
SYMBOL_PRICE_VOLATILITY	Volatilidad del precio en tanto por ciento	double
SYMBOL_PRICE_THEORETICAL	Precio teórico de la opción	double
SYMBOL_PRICE_DELTA	Delta de la opción/garantía. Muestra cuántas unidades cambiará el precio de una opción al cambiar el precio del activo básico en 1 unidad	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
SYMBOL_PRICE_THETA	Teta de la opción /garantía. Número de puntos que perderá diariamente el precio de una opción debido al deterioro temporal, es decir, conforme se acerque la fecha de expiración	double
SYMBOL_PRICE_GAMMA	Gamma de la opción /garantía. Muestra la velocidad de cambio de delta, es decir, con qué	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	rapidez o lentitud cambia el premio de la opción	
SYMBOL_PRICE_VEGA	Vega de la opción /garantía. Muestra el número de puntos que cambiará el precio de una opción al cambiar la volatilidad un 1%	double
SYMBOL_PRICE_RHO	Ro de la opción /garantía. Muestra la sensibilidad del precio teórico de una opción ante un cambio	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	de interés de un 1%	
SYMBOL_PRICE_OMEGA	Omega de la opción /garantía. Elasticidad de una opción: cambio porcentual relativo del precio de una opción ante el cambio porcentual del precio del activo base	double
SYMBOL_PRICE_SENSITIVITY	Sensibilidad de la opción /garantía. Muestra cuántos puntos deberá cambiar el precio del activo base	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	de una opción para que el precio de la opción cambie un punto	

Para la función SymbolInfoString()

ENUM_SYMBOL_INFO_STRING

Identificador	Descripción	Tipo de la propiedad
SYMBOL_BASIS	Nombre del activo base para el símbolo	string
SYMBOL_CATEGORY	Nombre de la categoría o sector al que pertenece el símbolo comercial	string
SYMBOL_COUNTRY	Nombre del país con el que se relaciona el instrumento financiero	string
SYMBOL_SECTOR_NAME	Sector económico al que pertenece el	string

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	instrumento financiero	
SYMBOL_INDUSTRY_NAME	Rama económica o tipo de industrial al que pertenece el instrumento financiero	string
SYMBOL_CURRENCY_BASE	Divisa básica del instrumento	string
SYMBOL_CURRENCY_PROFIT	Divisa de beneficio	string
SYMBOL_CURRENCY_MARGIN	Divisa de margen	string
SYMBOL_BANK	Fuente de cotización actual	string
SYMBOL_DESCRIPTION	Descripción literal del símbolo	string
SYMBOL_EXCHANGE	Nombre de la bolsa o plataforma comercial en la que se comercia	string

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	con el símbolo	
SYMBOL_FORMULA	Fórmula para construir el precio del instrumento personalizado. Si el nombre del instrumento financiero incluido en la fórmula empieza con una cifra o contiene un carácter especial ("", ".", "-", "&", "#", etc.), el nombre de dicho instrumento deberá colocarse entre comillas. • Símbolos	string

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	intético: "@ESU19" /EURCAD • Spreaddecalendario: "Si-9	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	.13"-"Si-6.13" • Índice del euro: 34.388057266*pow(EURU	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	SD,0.3155)*pow(EURGBP,0.3056)*pow(EURJPY,0.1891)	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	*pow(EURCHF,0.11113)*pow(EURSEK,0.07855)	
SYMBOL_ISIN	Nombre del símbolo comercial en el sistema de los códigos internaci	string

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	onales para identificación de valores — ISIN (International Securities Identification Number). El código internacional de identificación de un valores un código de 12 dígitos que contiene letras y cifras y que identifica de forma unívoca a un valor mobiliario a nivel internacional. La presencia de esta propiedad del símbolo se determina en el lado del servidor comercial.	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
SYMBOL_PAGE	Dirección de la página de internet con información sobre el símbolo. Esta dirección se representará en forma de enlace al mirar las propiedades del símbolo en el terminal	string
SYMBOL_PATH	Ruta en el árbol de símbolos	string

El gráfico de precio del símbolo se puede construir usando como base el precio Bid o Last. De la elección del precio para la construcción del gráfico depende cómo se forman y representan las barras en el terminal. Los posibles valores de la propiedad SYMBOL_CHART_MODE se muestran en la enumeración ENUM_SYMBOL_CHART_MODE

ENUM_SYMBOL_CHART_MODE

Identificador	Descripción
SYMBOL_CHART_MODE_BID	Las barras se construyen según los precios Bid
SYMBOL_CHART_MODE_LAST	Las barras se construyen según los precios Last

Para cada símbolo se puede indicar varios modos de plazo de vigencia (vencimiento) de las órdenes pendientes. Cada modo tiene su bandera; las banderas pueden ser combinadas mediante la operación del lógico **OR** (|), por ejemplo, SYMBOL_EXPIRATION_GTC|SYMBOL_EXPIRATION_SPECIFIED. Para comprobar si un modo en concreto está permitido para un instrumento, hay que comparar el resultado del lógico **AND** (&) con la bandera del modo.

© 2000-2025, MetaQuotes Ltd.

Si para el símbolo se especifica la bandera SYMBOL_EXPIRATION_SPECIFIED, entonces cuando se envía la orden pendiente, se puede indicar con precisión hasta que momento dicha orden está vigente.

Identificador	Valor	Descripción
SYMBOL_EXPIRATION_GTC	1	La orden está vigente sin restricción de tiempo hasta su explícita cancelación
SYMBOL_EXPIRATION_DAY	2	La orden está vigente hasta que se termine el día
SYMBOL_EXPIRATION_SPECIFIED	4	El plazo de vencimiento se indica en la orden
SYMBOL_EXPIRATION_SPECIFIED_DAY	8	El día de vencimiento se especifica en la orden

Ejemplo:

```
//+------------------------------------------------------------------+
//| comprueba si el modo especificado de vencimiento está permitido|
//+------------------------------------------------------------------+
bool IsExpirationTypeAllowed(string symbol,int exp_type)
  {
//--- obtenemos el valor de la propiedad que describe los modos de vencimiento permiti
   int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- devolvemos true, si el modo exp_type está permitido
   return((expiration & exp_type)==exp_type);
  }
```

Si la propiedad SYMBOL_EXPIRATION_MODE tiene el valor SYMBOL_EXPIRATION_GTC (la orden está activa hasta su cancelación), entonces el plazo de expiración de las órdenes pendientes y de los niveles establecidos de StopLoss/TakeProfit se indica de forma adicional con la ayuda de la enumeración ENUM_SYMBOL_ORDER_GTC_MODE.

ENUM_SYMBOL_ORDER_GTC_MODE

Identificador	Descripción
SYMBOL_ORDERS_GTC	Las órdenes pendientes y los niveles de Stop Loss/Take Profit son válidas por un tiempo ilimitado hasta que se cancelen de forma expresa
SYMBOL_ORDERS_DAILY	Las órdenes son válidas solo dentro de un día comercial. Tras su conclusión, todos los niveles de StopLoss y TakeProfit son eliminados, y también se eliminan las órdenes pendientes
SYMBOL_ORDERS_DAILY_EXCLUDING_STOPS	Al cambiar de día comercial, se eliminan solo las órdenes pendientes, los niveles de StopLoss y TakeProfit se conservan

© 2000-2025, MetaQuotes Ltd.

Al enviar una orden, se podrá especificar la política de rellenado para el volumen declarado en la orden comercial. Las opciones permitidas para la ejecución de órdenes por volumen para cada símbolo se muestran en el recuadro. Para cada instrumento, es posible configurar no solo un modo, sino varios, usando para ello una combinación de banderas. La combinación de banderas se expresa mediante la operación lógica **O** (|), por ejemplo, SYMBOL_FILLING_FOK|SYMBOL_FILLING_IOC. Para verificar si un modo específico está permitido para un instrumento, habrá que comparar el resultado de la **Y** lógica (&) con la bandera de modo: ejemplo.

Política de rellenado	Identificador	Valor	Descripción
Todo/Nada	SYMBOL_FILLING_FOK	1	La orden solo puede ser ejecutada en

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			el volumen indicado. Si en el mercado en estos

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			momentos no existe volumen suficiente de unin

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			strumento financiero, la orden no será ejecut

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			ada. El volumen necesario puede componerse de

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			variasofertasdisponiblesenelmercadoenel

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			momento actual. Para esta política, alenvia

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			runa orden, sedeberáindicarel tipoderellen

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			adoORDER_FILLING_FOK. La posibilidad de eus

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			arórdenesFOKsedeterminaenelservidorcome

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			rcial.
Todo/Parte	SYMBOL_FILLING_IOC	2	El trádera ceptar realizar una transacc

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			ión al volumen máximo disponible en el mercado

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			dentro del indicador en la orden. En caso de que

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			o sea posible la ejecución completa, la orden

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			ejecutará con el volumen disponible, mientr

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			asqueelvolumendelaorddennoejecutadoserác

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			ancelado. Para esta política, alenviar una o

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			rden, sedeberáindicarel tipoderellenado O R

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			DER_FILLING_IOC. La posibilidad de usar ór d

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			en el IOC se determina en el servidor comercia

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			l.
Pasiva	SYMBOL_FILLING_BOC	4	La política BOC (Book or Cancel) implica q

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			uelaordensolopuededecolocarseenlaProfundi

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			dadde Mercadoyno puedeejecutarseinmediat

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			amente. Si la orden puede ejecutarse inmedia

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			tamente alcolocarse, entonces se eliminará.

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			En la práctica, esta política de ejecución so

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			lo puede indicar se cuando el precio de la orden

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			colocada se aplee or que el del mercado o actual. La

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			sórdenesBoCseutilizanparaimplementarlan

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			egociaciónpasiva, deformaquesegaranticeq

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			uelaordenonoseejecutarárainmediatamentetra

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			s s u c o l o c a c i ó n y n o a f e c t e a l a l i q u i d e z a c t u a

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			l. Se admite solo para órdenes límite y stop límite

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			mite, es decir, si la banderaSYMBOL_ORDER_MO

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			DE contiene los valores SYMBOL_ORDER_LIMITy

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			/o SYMBOL_ORDER_STOP_LIMIT.
Devolver	No hay identificador		Si habla mos del

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			aejecuciónparcialpormercadoodeunaordenl

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			límite con volumen restante, no se equitará, sin

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			oqueconti nuarráactiva. Paraestapolítica,

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			al enviar una orden, se debe especificar el tip

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			o de rellenado ORDER_FILLING_RETURN. Lasór

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			denesRetornoestápermitidaselmodode

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			jecucciónMarketExecution(ejecucióndeórde

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			nesporermercado_SYMBOL_TRADE_EXECUTION_MA

© 2000-2025, MetaQuotes Ltd.

Política de rellenado	Identificador	Valor	Descripción
			RKET).

Al enviar una solicitud comercial usando la funciónOrderSend(), la política de ejecución necesaria por volumen se puede configurar en el campo *type_filling*, en la estructura especial MqlTradeRequest se permiten valores de la enumeración ENUM_ORDER_TYPE_FILLING. Si no se especifica el tipo de rellenado, este será automáticamente sustituido en la orden comercial por ORDER_FILLING_RETURN. En este caso, debemos notar que el tipo de rellenado ORDER_FILLING_RETURN está permitido en todos los modos de ejecución salvo el modo "Ejecución por mercado" (SYMBOL_TRADE_EXECUTION_MARKET).

Al enviar una solicitud comercial de ejecución **en el momento actual** (time in force), hay que considerar que no existe garantía en los mercados financieros de que en un momento dado esté disponible para este instrumento financiero todo el volumen solicitado al precio deseado. Por consiguiente, el comercio en tiempo real está regulado por los modos de ejecución de precio y volumen. Los modos o políticas de ejecución determinan las reglas para los casos en los que ha cambiado el precio, o el volumen solicitado no se puede ejecutar al completo **en el momento actual**.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
Modo de ejecución (Request Execution)	Ejecución de una orden de mercado al precio recibido previamente del bróker.	SYMBOL_TRADE_EXECUTION_REQUEST

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	Antes de enviar una orden de mercado, se le solicitarán al bróker sus precios de ejecución. Después de recibirlos, la ejecución de la orden a un precio determinado podrá confirmarse o rechazarse.	
Ejecución instantánea (Instant Execution)	Ejecución instantánea de una orden de mercado al precio indicado.	SYMBOL_TRADE_EXECUTION_INSTANT

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	Al enviar una solicitud comercial para su ejecución, la plataforma sustituirá automáticamente los precios actuales en la orden. • Si el bróker acepta el precio, la	

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	orden se ejecutará. • Si el bróker no acepta el precio solic	

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	itado, se produciráal allamada a "recotización" (Requote	

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	): el bróker retornará los precios a los que se puede ejecu	

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	tarestaordenen.	
Ejecución por mercado (Market Execution)	El bróker tomará la decisión sobre el precio de ejecución sin un acuerdo adicional con el tráder. Enviar una orden de mercado en este modo implica la aceptación anticipada del precio	SYMBOL_TRADE_EXECUTION_MARKET

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	al que se ejecutará.	
Ejecución por bolsa (Exchange Execution)	Las operaciones comerciales se realizan según los precios de las ofertas de mercado actuales.	SYMBOL_TRADE_EXECUTION_EXCHANGE

Antes de enviar una orden con ejecución en el momento actual, para establecer correctamente el valor ORDER_TYPE_FILLING (tipo de ejecución por volumen), podemos obtener para cada instrumento financiero con la ayuda de la función SymbolInfoInteger() el valor de la propiedad SYMBOL_FILLING_MODE, que muestra en forma de combinación de banderas los tipos de ejecución por volumen permitidos para este símbolo. Cabe señalar que el tipo de rellenado ORDER_FILLING_RETURN siempre está permitido, salvo para el modo "Ejecución por mercado" (SYMBOL_TRADE_EXECUTION_MARKET).

El uso de los tipos de rellenado según el modo de ejecución se puede representar en forma de recuadro:

Modo de ejecución\Política de rellenado	Todo/Nada (FOK ORDER_FILLING_FOK)	Todo/Parte (IOC ORDER_FILLING_IOC)	Devolver (Return ORDER_FILLING_RETURN)
Ejecución instantánea (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (independientemente de los ajustes del símbolo)	+ (independientemente de los ajustes del símbolo)	+ (siempre)
Ejecución por solicitud SYMBOL_TRADE_EXECUTION_REQUEST	+ (independientemente de los ajustes del símbolo)	+ (independientemente de los ajustes del símbolo)	+ (siempre)

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución\Política de rellenado	Todo/Nada (FOK ORDER_FILLING_FOK)	Todo/Parte (IOC ORDER_FILLING_IOC)	Devolver (Return ORDER_FILLING_RETURN)
Ejecución por mercado SYMBOL_TRADE_EXECUTION_MARKET	+ (se establece en los ajustes del símbolo)	+ (se establece en los ajustes del símbolo)	- (prohibido independientemente de los ajustes del símbolo)
Ejecución por bolsa SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (se establece en los ajustes del símbolo)	+ (se establece en los ajustes del símbolo)	+ (siempre)

Para las órdenes pendientes, independientemente del modo de ejecución (SYMBOL_TRADE_EXEMODE), se debe usar el tipo de rellenado ORDER_FILLING_RETURN, ya que dichas órdenes no están pensadas para ejecutarse en el momento del envío. Al usar órdenes pendientes, el tráder aceptará de antemano que cuando se den las condiciones para una transacción en esta orden, el bróker usará el tipo de rellenado que sea compatible con la plataforma comercial dada.

Ejemplo:

```
//+------------------------------------------------------------------+
//| comprobamos si el modo de ejecución indicado está permitido      |
//+------------------------------------------------------------------+
bool IsFillingTypeAllowed(string symbol,int fill_type)
  {
//--- obtenemos el valor de la propiedad que describe el modo de rellenado
   int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- retornamos true, si el modo fill_type está permitido
   return((filling&fill_type)==fill_type);
  }
```

Cuando se envía una solicitud comercial usando la función OrderSend(), para algunas operaciones hace falta especificar el tipo de la oreden desde la enumeración ENUM_ORDER_TYPE. No todos los tipos de órdenes pueden estar permitidos para un símbolo específico, la propiedad SYMBOL_ORDER_MODE describe las banderas de los tipos permitidos.

Identificador	Valor	Descripción
SYMBOL_ORDER_MARKET	1	Están permitidas las órdenes de mercado (Buy y Sell en el mercado sin especificar el precio de transacción)
SYMBOL_ORDER_LIMIT	2	Están permitidas las órdenes limitadas (Buy Limit y Sell Limit)

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor	Descripción
SYMBOL_ORDER_STOP	4	Están permitidas las órdenes Stop (Buy Stop y Sell Stop)
SYMBOL_ORDER_STOP_LIMIT	8	Están permitidas las órdenes Stop Limit (Buy Stop Limit y Sell Stop Limit)
SYMBOL_ORDER_SL	16	Está permitida la colocación de Stop Loss
SYMBOL_ORDER_TP	32	Está permitida la colocación de Take Profit
SYMBOL_ORDER_CLOSEBY	64	Autorización para cerrar la posición usando otra opuesta, es decir, usando una posición abierta en la dirección contraria, en el mismo instrumento. La propiedad es definida para cuentas con cobertura (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)

Ejemplo:

```
//+------------------------------------------------------------------+
//| La función imprime los tipos de órdenes permitidos para el símbolo|
//+------------------------------------------------------------------+
void Check_SYMBOL_ORDER_MODE(string symbol)
  {
//--- obtenemos el valor de la propiedad que describe los tipos de órdenes permitidos
   int symbol_order_mode=(int)SymbolInfoInteger(symbol,SYMBOL_ORDER_MODE);
//--- chequeo para órdenes de mercado (Market Execution)
   if((SYMBOL_ORDER_MARKET&symbol_order_mode)==SYMBOL_ORDER_MARKET)
      Print(symbol+": Las órdenes de mercado están permitidas (no se requiere indicar
//--- chequeo para órdenes del tipo Limit
   if((SYMBOL_ORDER_LIMIT&symbol_order_mode)==SYMBOL_ORDER_LIMIT)
      Print(symbol+": Las órdenes Buy Limit y Sell Limit están permitidas");
//--- chequeo para órdenes del tipo Stop
   if((SYMBOL_ORDER_STOP&symbol_order_mode)==SYMBOL_ORDER_STOP)
      Print(symbol+": Las órdenes Buy Stop y Sell Stop están permitidas");
//--- chequeo para órdenes del tipo Stop Limit
   if((SYMBOL_ORDER_STOP_LIMIT&symbol_order_mode)==SYMBOL_ORDER_STOP_LIMIT)
      Print(symbol+": Las órdenes Buy Stop Limit y Sell Stop Limit están permitidas");
//--- comprobación de posibilidad de colocación de las órdenes Stop Loss
   if((SYMBOL_ORDER_SL&symbol_order_mode)==SYMBOL_ORDER_SL)
      Print(symbol+": Las órdenes Stop Loss están permitidas");
//--- comprobación de posibilidad de colocación de las órdenes Take Profit
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_TP)
      Print(symbol+": Las órdenes Take Profit están permitidas");
//---
  }
```

La enumeración ENUM_SYMBOL_CALC_MODE sirve para obtener la información sobre el cálculo del monto de los fondos prendarios (monto de los requerimientos de margen).

© 2000-2025, MetaQuotes Ltd.

ENUM_SYMBOL_CALC_MODE

Identificador	Descripción	Formula
SYMBOL_CALC_MODE_FOREX	Forex mode - cálculo de beneficio y margen para Forex	Margin: Lots * Contract_Size / Leverage * Margin_Rate Profit: (close_price - open_price) * Contract_Size*Lots
SYMBOL_CALC_MODE_FOREX_NO_LEVERAGE	Forex No Leverage mode - cálculo del beneficio y el margen para símbolos Fórex sin tener en cuenta el apalancamiento	Margin: Lots * Contract_Size * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_FUTURES	Futures mode - cálcu	Margin: Lots * InitialMargin * Margin_Rate Profit: (close_price - open_price) * TickPrice / TickSize*Lots

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	lo de beneficio y margen para futuros	
SYMBOL_CALC_MODE_CFD	CFD mode - cálculo de beneficio y margen para CFD	Margin: Lots * ContractSize * MarketPrice * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_CFDINDEX	CFD index mode - cálculo de beneficio y margen para CFD por índices	Margin: (Lots * ContractSize * MarketPrice) * TickPrice / TickSize * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD Leverage mode - cálculo de	Margin: (Lots * ContractSize * MarketPrice) / Leverage * Margin_Rate Profit: (close_price-open_price) * Contract_Size * Lots

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	beneficio y margen para CFD en caso del trading con apalancamiento financiero	
SYMBOL_CALC_MODE_EXCH_STOCKS	Exchange mode - cálculo de beneficio y margen para tradear con los valores en la bolsa	Margin: Lots * ContractSize * LastPrice * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_FUTURES	Futures mode - cálculo de beneficio	Margin: Lots * InitialMargin * Margin_Rate o Lots * MaintenanceMargin * Margin_Rate Profit: (close_price - open_price) * Lots * TickPrice / TickSize

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	y margen para tradear con los contratos de futuros en la bolsa	
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS Futures mode - cálculo de beneficio y margen para tradear con los contratos de futuros en FORTS. El margen puede reducirse	Margin: Lots * InitialMargin * Margin_Rate o Lots * MaintenanceMargin * Margin_Rate Profit: (close_price - open_price) * Lots * TickPrice / TickSize

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	por el tamaño de la desviación MarginDiscount según las siguientes reglas: 1 Si el precio de la posición larga (orden de compra) es inferior al precio de liquidación, entonces MarginDiscount = Lots*((PriceSe	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	ttle-PriceOrder) *TickPrice/TickSize) 2. Si el precio de la posición corta (orden de venta) es superior al precio de liquidación, entonces MarginDiscount = Lots*((PriceOrder-PriceSettle)*TickPrice/TickSize)	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	donde: ○ PriceSettle - precio de liquidación (clearing) de la	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	sesión anterior; ○ Price Order-precio medio ponderado	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	elaposicióndeapertura especificadoenlaord	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	en (solicitud); ○ TickPrice - precio del tick (costeda va	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	riación del precio por un punto) TickSize - tamaño del ti	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	ck(pasomínimodelcambiodelprecio)	
SYMBOL_CALC_MODE_EXCH_BONDS	Exchange Bonds mode - cálculo del margen y el beneficio para el	Margin: Lots * ContractSize * FaceValue * open_price * /100 Profit: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	comercio con obligaciones en la bolsa	
SYMBOL_CALC_MODE_EXCH_STOCKS_MOEX	Exchange MOEX Stocks mode - cálculo del margen y el beneficio para el comercio con valores en la bolsa MOEX	Margin: Lots * ContractSize * LastPrice * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_BONDS_MOEX	Exchange MOEX Bonds mode - cálculo del margen y	Margin: Lots * ContractSize * FaceValue * open_price * /100 Profit: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	el beneficio para el comercio con obligaciones en la bolsa MOEX	
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - el símbolo se usa como un activo no tradeable en la cuenta comercial. El valor de mercado de la posición abierta se calcula a base	Margin: no hay Profit: no hay Valor de mercado: Lots * ContractSize * MarketPrice * LiqudityRate

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	del volumen, precio actual de mercado, tamaño del contrato y el ratio de la liquidez. Este valor se incluye en los Activos (Assets) que se suman a la Equidad (Equity). De este modo, las posiciones abiertas	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Formula
	de este instrumento aumentan el tamaño del Margen Libre (Free Margin) y sirven de garantía adicional para las posiciones abiertas de los instrumentos tradeables.	

Existen varias formas de tradear con el instrumento. La información sobre los regímenes de comerciar con cada instrumento en concreto se especifica en los valores de la enumeración ENUM_SYMBOL_TRADE_MODE.

ENUM_SYMBOL_TRADE_MODE

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
SYMBOL_TRADE_MODE_DISABLED	Trading por el símbolo prohibido
SYMBOL_TRADE_MODE_LONGONLY	Sólo compras
SYMBOL_TRADE_MODE_SHORTONLY	Sólo ventas
SYMBOL_TRADE_MODE_CLOSEONLY	Permitidas sólo operaciones de cierre de posiciones
SYMBOL_TRADE_MODE_FULL	Sin restricciones de las operaciones comerciales

En la enumeración ENUM_SYMBOL_TRADE_EXECUTION se especifican los posibles regímenes de llevar a cabo las transacciones por cada instrumento en concreto.

ENUM_SYMBOL_TRADE_EXECUTION

Identificador	Descripción
SYMBOL_TRADE_EXECUTION_REQUEST	Ejecución por Pedido
SYMBOL_TRADE_EXECUTION_INSTANT	Ejecución Instantánea
SYMBOL_TRADE_EXECUTION_MARKET	Ejecución de órdenes por Mercado
SYMBOL_TRADE_EXECUTION_EXCHANGE	Ejecución por Bolsa

Los modos de calcular los swaps al cambiar de posición se especifican el la enumeración ENUM_SYMBOL_SWAP_MODE. El modo de calcular los swaps determina las unidades de medición de lsos parámetros SYMBOL_SWAP_LONG y SYMBOL_SWAP_SHORT. Por ejemplo, si los swaps se calculan en la divisa del depósito del cliente, en los parámetros el volumen de los swaps calculados se indica precisamente en la divisa del depósito del cliente.

ENUM_SYMBOL_SWAP_MODE

Identificador	Descripción
SYMBOL_SWAP_MODE_DISABLED	No hay swaps
SYMBOL_SWAP_MODE_POINTS	Swaps se calculan en puntos
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Swaps se calculan en dinero en divisa base del símbolo
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Swaps se calculan en dinero en divisa marginal del símbolo
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Swaps se calculan en dinero en divisa del depósito del cliente
SYMBOL_SWAP_MODE_CURRENCY_PROFIT	Swaps se calculan en dinero en la divisa de cálculo de los beneficios

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Swaps se calculan en por cientos anuales del precio del instrumento para el momento de cálculo del swap (régimen bancario es de 360 días al año)
SYMBOL_SWAP_MODE_INTEREST_OPEN	Swaps se calculan en por cientos anuales del precio de apertura de la posición para el símbolo (régimen bancario es de 360 días al año)
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Swaps se calculan por reapertura de posiciones. Al final de la sesión de trading la posición se cierra forzosamente. Al día siguiente la posición vuelve a abrirse por el precio de cierre +/- el número de puntos especificado (en los parámetros SYMBOL_SWAP_LONG y SYMBOL_SWAP_SHORT)
SYMBOL_SWAP_MODE_REOPEN_BID	Swaps se calculan por reapertura de posiciones. Al final de la sesión de trading la posición se cierra forzosamente. Al día siguiente la posición vuelve a abrirse por el precio actual Bid +/- el número de puntos especificado (en los parámetros SYMBOL_SWAP_LONG y SYMBOL_SWAP_SHORT)

La enumeración ENUM_DAY_OF_WEEK sirve para indicar el día de la semana.

ENUM_DAY_OF_WEEK

Identificador	Descripción
SUNDAY	Domingo
MONDAY	Lunes
TUESDAY	Martes
WEDNESDAY	Miércoles
THURSDAY	Jueves
FRIDAY	Viernes
SATURDAY	Sábado

La opción es un contrato que da derecho pero no obligación para comprar o vender el activo base (bienes, acciones, futuros, etc) por un precio fijo durante la vigencia de la opción o en un momento de tiempo determinado. Para la descripción de las propiedades de las opciones se usan las enumeraciones que describen el tipo de la opción y el derecho que concede.

© 2000-2025, MetaQuotes Ltd.

ENUM_SYMBOL_OPTION_RIGHT

Identificador	Descripción
SYMBOL_OPTION_RIGHT_CALL	La opción que concede el derecho a comprar el activo por el precio fijo
SYMBOL_OPTION_RIGHT_PUT	La opción que concede el derecho a vender el activo por el precio fijo

ENUM_SYMBOL_OPTION_MODE

Identificador	Descripción
SYMBOL_OPTION_MODE_AMERICAN	Tipo europeo de la opción - puede ser amortizado sólo en la fecha especificada (fecha de vencimiento del plazo, fecha de ejecución, fecha de reembolso)
SYMBOL_OPTION_MODE_EUROPEAN	Tipo americano de la opción - puede ser amortizado en cualquier fecha antes del vencimiento de la opción. Para este tipo se establece un período durante el cual el comprador puede ejecutar esta opción

Los instrumentos financieros se distribuyen por sectores económicos. El sector económico se determina por las características generales, los objetivos económicos, las funciones y comportamientos, lo cuales permiten distinguir este sector de otras partes de la economía. En ENUM_SYMBOL_SECTOR, se enumeran los sectores de la economía a los que puede pertenecer un instrumento comercial.

ENUM_SYMBOL_SECTOR

Identificador	Descripción
SECTOR_UNDEFINED	No definido
SECTOR_BASIC_MATERIALS	Materias primas
SECTOR_COMMUNICATION_SERVICES	Servicios de comunicación
SECTOR_CONSUMER_CYCLICAL	Consumo de demanda cíclica
SECTOR_CONSUMER_DEFENSIVE	Consumo básico
SECTOR_CURRENCY	Divisas
SECTOR_CURRENCY_CRYPTO	Criptodivisas
SECTOR_ENERGY	Energía
SECTOR_FINANCIAL	Finanzas
SECTOR_HEALTHCARE	Sanidad

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
SECTOR_INDUSTRIALS	Industria
SECTOR_REAL_ESTATE	Bienes inmuebles
SECTOR_TECHNOLOGY	Tecnologías
SECTOR_UTILITIES	Servicios residenciales
SECTOR_INDEXES	Índices
SECTOR_COMMODITIES	Productos bursátiles

Cada instrumento financiero puede pertenecer a una industria o sector concretos. Entendemos por sector de la economía el conjunto de empresas que generan una producción homogénea o específica con ayuda de tecnologías de un mismo tipo. En ENUM_SYMBOL_INDUSTRY, se enumeran los tipos de industria a los que puede pertenecer un instrumento comercial.

ENUM_SYMBOL_INDUSTRY

Identificador	Descripción
INDUSTRY_UNDEFINED	No definido
Materias primas	
INDUSTRY_AGRICULTURAL_INPUTS	Recursos agrarios
INDUSTRY_ALUMINIUM	Aluminio
INDUSTRY_BUILDING_MATERIALS	Materiales de construcción
INDUSTRY_CHEMICALS	Productos químicos
INDUSTRY_COKING_COAL	Carbón de coque
INDUSTRY_COPPER	Cobre
INDUSTRY_GOLD	Oro
INDUSTRY_LUMBER_WOOD	Madera y productos derivados
INDUSTRY_INDUSTRIAL_METALS	Metales industriales
INDUSTRY_PRECIOUS_METALS	Metales preciosos
INDUSTRY_PAPER	Papel y productos derivados de la celulosa
INDUSTRY_SILVER	Plata
INDUSTRY_SPECIALTY_CHEMICALS	Productos químicos especiales
INDUSTRY_STEEL	Acero
Servicios de comunicación	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
INDUSTRY_ADVERTISING	Agencias publicitarias
INDUSTRY_BROADCASTING	Radiodifusión y televisión
INDUSTRY_GAMING_MULTIMEDIA	Juegos electrónicos y multimedia
INDUSTRY_ENTERTAINMENT	Entretenimiento
INDUSTRY_INTERNET_CONTENT	Contenido de internet e información
INDUSTRY_PUBLISHING	Editoriales
INDUSTRY_TELECOM	Servicios de telecomunicación
Consumo de demanda cíclica	
INDUSTRY_APPAREL_MANUFACTURING	Manufactura de ropa
INDUSTRY_APPAREL_RETAIL	Venta minorista de ropa
INDUSTRY_AUTO_MANUFACTURERS	Fabricación de automóviles
INDUSTRY_AUTO_PARTS	Repuestos de automóvil
INDUSTRY_AUTO_DEALERSHIP	Concesionarios de automóviles
INDUSTRY_DEPARTMENT_STORES	Grandes almacenes
INDUSTRY_FOOTWEAR_ACCESSORIES	Calzado y accesorios
INDUSTRY_FURNISHINGS	Muebles y electrodomésticos
INDUSTRY_GAMBLING	Juegos de azar
INDUSTRY_HOME_IMPROV_RETAIL	Comercio minorista de productos para el hogar
INDUSTRY_INTERNET_RETAIL	Comercio minorista online
INDUSTRY_LEISURE	Ocio
INDUSTRY_LODGING	Vivienda
INDUSTRY_LUXURY_GOODS	Productos de lujo
INDUSTRY_PACKAGING_CONTAINERS	Embalajes
INDUSTRY_PERSONAL_SERVICES	Servicios personales
INDUSTRY_RECREATIONAL_VEHICLES	Medios de transporte para el ocio
INDUSTRY_RESIDENT_CONSTRUCTION	Construcción de viviendas
INDUSTRY_RESORTS_CASINOS	Resorts y casinos
INDUSTRY_RESTAURANTS	Restaurantes
INDUSTRY_SPECIALTY_RETAIL	Comercio minorista especializado
INDUSTRY_TEXTILE_MANUFACTURING	Producción textil

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
INDUSTRY_TRAVEL_SERVICES	Servicios de viajes
Consumo básico	
INDUSTRY_BEVERAGES_BREWERS	Bebidas alcohólicas - Cerveceras
INDUSTRY_BEVERAGES_NON_ALCO	Bebidas - Bebidas sin alcohol
INDUSTRY_BEVERAGES_WINERIES	Bebidas - Bodegas y fábricas de licores
INDUSTRY_CONFECTIONERS	Pastelería y confitería
INDUSTRY_DISCOUNT_STORES	Tiedas de descuentos
INDUSTRY_EDUCATION_TRAINIG	Educación y entrenamiento
INDUSTRY_FARM_PRODUCTS	Productos agrícolas
INDUSTRY_FOOD_DISTRIBUTION	Distribución de productos alimenticios
INDUSTRY_GROCERY_STORES	Supermercados
INDUSTRY_HOUSEHOLD_PRODUCTS	Productos para el hogar
INDUSTRY_PACKAGED_FOODS	Embalaje de productos
INDUSTRY_TOBACCO	Tabaco
Energía	
INDUSTRY_OIL_GAS_DRILLING	Extracción de petróleo y gas
INDUSTRY_OIL_GAS_EP	Extracción y procesamiento de petróleo y gas
INDUSTRY_OIL_GAS_EQUIPMENT	Equipamiento y servicios de petróleo y gas
INDUSTRY_OIL_GAS_INTEGRATED	Compañías de petróleo y gas integradas
INDUSTRY_OIL_GAS_MIDSTREAM	Transporte de petróleo y gas
INDUSTRY_OIL_GAS_REFINING	Refinado de petróleo y gas
INDUSTRY_THERMAL_COAL	Carbón térmico
INDUSTRY_URANIUM	Uranio
Finanzas	
INDUSTRY_EXCHANGE_TRADED_FUND	Fondo bursátil
INDUSTRY_ASSETS_MANAGEMENT	Gestión de activos
INDUSTRY_BANKS_DIVERSIFIED	Bancos - Diversificados
INDUSTRY_BANKS_REGIONAL	Bancos - Regionales
INDUSTRY_CAPITAL_MARKETS	Mercados financieros
INDUSTRY_CLOSE_END_FUND_DEBT	Fondo de capital fijo - Instrumentos de deuda

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
INDUSTRY_CLOSE_END_FUND_EQUITY	Fondo de capital fijo - Acciones
INDUSTRY_CLOSE_END_FUND_FOREIGN	Fondo de capital fijo - Extranjero
INDUSTRY_CREDIT_SERVICES	Servicios crediticios
INDUSTRY_FINANCIAL_CONGLOMERATE	Conglomerados financieros
INDUSTRY_FINANCIAL_DATA_EXCHANGE	Datos financieros y bolsa
INDUSTRY_INSURANCE_BROKERS	Corredor de seguros
INDUSTRY_INSURANCE_DIVERSIFIED	Seguros - Diversificados
INDUSTRY_INSURANCE_LIFE	Seguros - Seguros de vida
INDUSTRY_INSURANCE_PROPERTY	Seguros - Inmuebles y accidentes
INDUSTRY_INSURANCE_REINSURANCE	Seguros - Reaseguros
INDUSTRY_INSURANCE_SPECIALTY	Seguros - Especiales
INDUSTRY_MORTGAGE_FINANCE	Financiación hipotecaria
INDUSTRY_SHELL_COMPANIES	Empresas fantasma
Sanidad	
INDUSTRY_BIOTECHNOLOGY	Biotecnología
INDUSTRY_DIAGNOSTICS_RESEARCH	Diagnóstico e investigación
INDUSTRY_DRUGS_MANUFACTURERS	Producción de medicamentos - General
INDUSTRY_DRUGS_MANUFACTURERS_SPEC	Producción de medicamentos - Especial
INDUSTRY_HEALTHCARE_PLANS	Planes de sanidad
INDUSTRY_HEALTH_INFORMATION	Servicios informativos de la salud
INDUSTRY_MEDICAL_FACILITIES	Centros médicos
INDUSTRY_MEDICAL_DEVICES	Equipamiento médico
INDUSTRY_MEDICAL_DISTRIBUTION	Distribuidores médicos
INDUSTRY_MEDICAL_INSTRUMENTS	Instrumental médico y materiales desechables
INDUSTRY_PHARM_RETAILERS	Minoristas farmacéuticos
Industria	
INDUSTRY_AEROSPACE_DEFENSE	Industria aeroespacial y de defensa
INDUSTRY_AIRLINES	Compañías aéreas
INDUSTRY_AIRPORTS_SERVICES	Aeropuertos y transporte aéreo
INDUSTRY_BUILDING_PRODUCTS	Materiales y equipamiento de construcción

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
INDUSTRY_BUSINESS_EQUIPMENT	Equipamiento y materiales de oficina
INDUSTRY_CONGLOMERATES	Conglomerados
INDUSTRY_CONSULTING_SERVICES	Servicios de consulting
INDUSTRY_ELECTRICAL_EQUIPMENT	Equipamiento y materiales eléctricos
INDUSTRY_ENGINEERING_CONSTRUCTION	Ingeniería y construcción
INDUSTRY_FARM_HEAVY_MACHINERY	Maquinaria pesada agraria y de la construcción
INDUSTRY_INDUSTRIAL_DISTRIBUTION	Distribuidores industriales
INDUSTRY_INFRASTRUCTURE_OPERATIONS	Operaciones de infraestructura
INDUSTRY_FREIGHT_LOGISTICS	Transporte y logística integrados
INDUSTRY_MARINE_SHIPPING	Transporte marítimo
INDUSTRY_METAL_FABRICATION	Fabricación de metales
INDUSTRY_POLLUTION_CONTROL	Control de contaminación
INDUSTRY_RAILROADS	Ferrocarriles
INDUSTRY_RENTAL_LEASING	Alquiler y arrendamiento
INDUSTRY_SECURITY_PROTECTION	Seguridad y protección
INDUSTRY_SPEALITY_BUSINESS_SERVICES	Servicios financieros especializados
INDUSTRY_SPEALITY_MACHINERY	Maquinaria industrial especializada
INDUSTRY_STUFFING_EMPLOYMENT	Servicios de empleo
INDUSTRY_TOOLS_ACCESSORIES	Herramientas e inventarios
INDUSTRY_TRUCKING	Transporte de cargas
INDUSTRY_WASTE_MANAGEMENT	Gestión de residuos
Bienes inmuebles	
INDUSTRY_REAL_ESTATE_DEVELOPMENT	Inmuebles - Construcción
INDUSTRY_REAL_ESTATE_DIVERSIFIED	Inmuebles - Diversificados
INDUSTRY_REAL_ESTATE_SERVICES	Servicios en el sector inmobiliario
INDUSTRY_REIT_DIVERSIFIED	Fondo de inversión inmobiliaria - Diversificados
INDUSTRY_REIT_HEALTCARE	Fondo de inversión inmobiliaria - Centros médicos
INDUSTRY_REIT_HOTEL_MOTEL	Fondo de inversión inmobiliaria - Hoteles
INDUSTRY_REIT_INDUSTRIAL	Fondo de inversión inmobiliaria - Industria
INDUSTRY_REIT_MORTAGE	Fondo de inversión inmobiliaria - Hipotecas

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
INDUSTRY_REIT_OFFICE	Fondo de inversión inmobiliaria - Oficinas
INDUSTRY_REIT_RESIDENTAL	Fondo de inversión inmobiliaria - Vivienda
INDUSTRY_REIT_RETAIL	Fondo de inversión inmobiliaria - Minoristas
INDUSTRY_REIT_SPECIALITY	Fondo de inversión inmobiliaria - Instalaciones especiales
Tecnologías	
INDUSTRY_COMMUNICATION_EQUIPMENT	Equipamiento de comunicación
INDUSTRY_COMPUTER_HARDWARE	Equipamiento de computadoras
INDUSTRY_CONSUMER_ELECTRONICS	Electrónica de consumo
INDUSTRY_ELECTRONIC_COMPONENTS	Componentes electrónicos
INDUSTRY_ELECTRONIC_DISTRIBUTION	Distribución electrónica y de computadoras
INDUSTRY_IT_SERVICES	Servicios de tecnologías de la información
INDUSTRY_SCIENTIFIC_INSTRUMENTS	Instrumentos científico-tecnológicos
INDUSTRY_SEMICONDUCTOR_EQUIPMENT	Equipamiento y materiales semiconductores
INDUSTRY_SEMICONDUCTORS	Semiconductores
INDUSTRY_SOFTWARE_APPLICATION	Software - Aplicaciones
INDUSTRY_SOFTWARE_INFRASTRUCTURE	Software - Infraestructura
INDUSTRY_SOLAR	Energía solar
Servicios residenciales	
INDUSTRY_UTILITIES_DIVERSIFIED	Empresas de servicios residenciales - Diversificados
INDUSTRY_UTILITIES_POWERPRODUCERS	Empresas de servicios residenciales - Productores independientes de energía
INDUSTRY_UTILITIES_RENEWABLE	Empresas de servicios residenciales - Energía renovable
INDUSTRY_UTILITIES_REGULATED_ELECTRIC	Empresas de servicios residenciales - Compañías eléctricas reguladas
INDUSTRY_UTILITIES_REGULATED_GAS	Empresas de servicios residenciales - Compañías de gas reguladas
INDUSTRY_UTILITIES_REGULATED_WATER	Empresas de servicios residenciales - Compañías de agua reguladas

© 2000-2025, MetaQuotes Ltd.

Información sobre la cuenta

Las funciones AccountInfoInteger(), AccountInfoDouble() y AccountInfoString() sirven para obtener la información sobre la cuenta corriente. Los valores de parámetros de estas funciones aceptan los valores de las correspondientes enumeraciones ENUM_ACCOUNT_INFO.

Para la función AccountInfoInteger()

ENUM_ACCOUNT_INFO_INTEGER

Identificador	Descripción	Tipo de la propiedad
ACCOUNT_LOGIN	Número de cuenta	long
ACCOUNT_TRADE_MODE	Account trade mode	ENUM_ACCOUNT_TRADE_MODE
ACCOUNT_LEVERAGE	Cuantía de apalancamiento concedido	long
ACCOUNT_LIMIT_ORDERS	El número máximo permitido de las órdenes pendientes activas	int
ACCOUNT_MARGIN_SO_MODE	Modo de establecimiento del nivel mínimo permitido de	ENUM_ACCOUNT_STOPOUT_MODE

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	la margen	
ACCOUNT_TRADE_ALLOWED	Permiso del trading para la cuenta corriente	bool
ACCOUNT_TRADE_EXPERT	Permiso del trading para un Asesor Experto	bool
ACCOUNT_MARGIN_MODE	Modo de cálculo del margen	ENUM_ACCOUNT_MARGIN_MODE
ACCOUNT_CURRENCY_DIGITS	Número de decimales - para la moneda de la cuenta - requerido para mostrar con precisión los resultados comerciales	int

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
ACCOUNT_FIFO_CLOSE	Indica que la posición solo se puede cerrar por el principio FIFO. Si el valor de la propiedad es true, solo se permitirá cerrar la posición de cada símbolo de acuerdo con el orden en que se abrieron, es decir, primero, la más antigua, después, la más reciente, etc. Al intenta	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	r cerrar posiciones en un orden diferente, aparecerá un error. Para las cuentas sin el registro de posiciones con la ayuda de hedge (ACCOUNT_MARGIN_MODE!=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), la propiedad sempre será false	
ACCOUNT_HEDGE_ALLOWED	Indica que están permitidas las posicio	bool

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	nes opuestas para el símbolo	

Para la función AccountInfoDouble()

ENUM_ACCOUNT_INFO_DOUBLE

Identificador	Descripción	Tipo de la propiedad
ACCOUNT_BALANCE	Balance de la cuenta en divisa del depósito	double
ACCOUNT_CREDIT	Cuantía de crédito concedido en divisa del depósito	double
ACCOUNT_PROFIT	Cuantía del beneficio actual en la cuenta en divisa del depósito	double
ACCOUNT_EQUITY	Cuantía de fondos propios en la cuenta en divisa del depósito	double
ACCOUNT_MARGIN	Cuantía de margen reservada en la cuenta en	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	divisa del depósito	
ACCOUNT_MARGIN_FREE	Cuantía de fondos disponibles en la cuenta en divisa del depósito para la apertura de una posición	double
ACCOUNT_MARGIN_LEVEL	Nivel de margen en la cuenta en por cientos	double
ACCOUNT_MARGIN_SO_CALL	Nivel de margen que requiere el recargo de la cuenta. Dependiendo del ACCOUNT_MARGIN_SO_MODE establecido, éste va en por cientos o en divisa del depósito	double
ACCOUNT_MARGIN_SO_SO	Nivel de margen; al llegar a este nivel, la posición menos rentable se cierra automátic	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	amente. Dependiendo del ACCOUNT_MARGIN_SO_MODE establecido, éste va en por cientos o en divisa del depósito	
ACCOUNT_MARGIN_INITIAL	Cuantía de fondos reservados en la cuenta para cubrir la garantía de todas las órdenes pendientes	double
ACCOUNT_MARGIN_MAINTENANCE	Cuantía de fondos reservados en la cuenta para cubrir el importe mínimo de todas las posiciones abiertas	double
ACCOUNT_ASSETS	Tamaño actual de fondos en la cuenta	double
ACCOUNT_LIABILITIES	Tamaño actual de obligacion	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo de la propiedad
	es en la cuenta	
ACCOUNT_COMMISSION_BLOCKED	Importe actual de comisiones bloqueadas de la cuenta	double

Para la función AccountInfoString()

ENUM_ACCOUNT_INFO_STRING

Identificador	Descripción	Tipo de la propiedad
ACCOUNT_NAME	Nombre de cliente	string
ACCOUNT_SERVER	Nombre del servidor comercial	string
ACCOUNT_CURRENCY	Divisa de depósito	string
ACCOUNT_COMPANY	Nombre de la empresa que atiende la cuenta	string

Hay varios tipos de cuentas que pueden estar abiertos en el servidor comercial. Para averiguar en qué tipo de cuenta opera un programa MQL5, se utiliza la enumeración ENUM_ACCOUNT_TRADE_MODE.

ENUM_ACCOUNT_TRADE_MODE

Identificador	Descripción
ACCOUNT_TRADE_MODE_DEMO	Cuenta comercial de demostración (Demo)
ACCOUNT_TRADE_MODE_CONTEST	Cuenta comercial de concurso (Contest)
ACCOUNT_TRADE_MODE_REAL	Cuenta comercial real (Real)

La situación del cierre forzoso Stop Out surge cuando faltan propios fondos para mantener las posiciones abiertas. El nivel mínimo de la margen que provoca Stop Out se puede establecer en por

© 2000-2025, MetaQuotes Ltd.

cientos o en dinero. La enumeración ENUM_ACCOUNT_STOPOUT_MODE sirve para averiguar el modo establecido para una cuenta en cuestión.

ENUM_ACCOUNT_STOPOUT_MODE

Identificador	Descripción
ACCOUNT_STOPOUT_MODE_PERCENT	Nivel en por cientos
ACCOUNT_STOPOUT_MODE_MONEY	Nivel en dinero

ENUM_ACCOUNT_MARGIN_MODE

Identifier	Description
ACCOUNT_MARGIN_MODE_RETAIL_NETTING	Se utiliza para el mercado no bursátil al registrar las posiciones en el modo "compensación" (en un símbolo puede haber solo una posición). El cálculo del margen se realiza basándose en el tipo de instrumento (SYMBOL_TRADE_CALC_MODE).
ACCOUNT_MARGIN_MODE_EXCHANGE	Se usa para el mercado bursátil. El cálculo del margen se realiza en base a los descuentos indicados en los ajustes de los instrumentos. Los descuentos son establecidos por el bróker, aunque no pueden ser inferiores a los valores determinados por la bolsa.
ACCOUNT_MARGIN_MODE_RETAIL_HEDGING	Se usan para el mercado no bursátil al realizarse el registro independiente de posiciones ("cobertura", en un símbolo pueden existir varias posiciones). El cálculo del margen se realiza basándose en el tipo de instrumento (SYMBOL_TRADE_CALC_MODE) y teniendo en cuenta el tamaño del margen cubierto (SYMBOL_MARGIN_HEDGED)..

Un ejemplo del script que muestra la información breve sobre la cuenta.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nombre de la empresa
   string company=AccountInfoString(ACCOUNT_COMPANY);
//--- nombre del cliente
   string name=AccountInfoString(ACCOUNT_NAME);
//--- número de la cuenta
   long login=AccountInfoInteger(ACCOUNT_LOGIN);
//--- nombre del servidor
   string server=AccountInfoString(ACCOUNT_SERVER);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- divisa de la cuenta
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- cuenta de demostración, de concurso o real
   ENUM_ACCOUNT_TRADE_MODE account_type=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(AC
//--- ahora transformaremos el valor de la enumeración en una forma comprensible
   string trade_mode;
   switch(account_type)
     {
      case  ACCOUNT_TRADE_MODE_DEMO:
         trade_mode="demo";
         break;
      case  ACCOUNT_TRADE_MODE_CONTEST:
         trade_mode="de concurso";
         break;
      default:
         trade_mode="real";
         break;
     }
//--- Stop Out se establece en por cientos o dinero
   ENUM_ACCOUNT_STOPOUT_MODE stop_out_mode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoInteg
//--- obtenemos valores de niveles cuando ocurren Margin Call y Stop Out
   double margin_call=AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL);
   double stop_out=AccountInfoDouble(ACCOUNT_MARGIN_SO_SO);
//--- visualizaremos la información breve sobre la cuenta
   PrintFormat("La cuenta del cliente '%s' #%d %s está abierta en '%s' en el servidor
               name,login,trade_mode,company,server);
   PrintFormat("Moneda de la cuenta - %s, el nivel MarginCall y StopOut se establece e
               currency,(stop_out_mode==ACCOUNT_STOPOUT_MODE_PERCENT)?"en por cientos"
   PrintFormat("Nivel MarginCall=%G, nivel StopOut=%G",margin_call,stop_out);
  }
```

© 2000-2025, MetaQuotes Ltd.

Estadística de simulación

Una vez finalizado el proceso de simulación, se calculan los indicadores estadísticos de los resultados comerciales según diferentes parámetros. Los valores de estos indicadores se puede conseguir a través de la función TesterStatistics(), especificando previamente el identificador del indicador desde de la enumeración ENUM_STATISTICS.

Aunque para calcular los datos estadísticos se utilizan dos tipos de parámetros - int y double - la función devuelve todos los valores como double. Por defecto, todos los valores estadísticos del tipo double se expresan en la moneda del depósito, si no está especificado lo otro.

ENUM_STATISTICS

Identificador	Descripción del parámetro estadístico	Tipo
STAT_INITIAL_DEPOSIT	Valor del depósito inicial	double
STAT_WITHDRAWAL	Fondos retirados de la cuenta	double
STAT_PROFIT	Beneficio neto tras la simulación, suma de STAT_GROSS_PROFIT y STAT_GROSS_LOSS (STAT_GROSS_LOSS siempre es menos o igual a cero)	double
STAT_GROSS_PROFIT	Beneficio bruto, importe de todas las transacciones rentables (con resultados positivos). Su valor es superior o igual a cero	double
STAT_GROSS_LOSS	Pérdidas brutas, importe de todas las transacciones de pérdidas (con resultados negativos). Su valor es inferior o igual a cero	double
STAT_MAX_PROFITTRADE	Beneficio máximo – el valor máximo entre todas las transacciones rentables. Su valor es superior o igual a cero	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción del parámetro estadístico	Tipo
STAT_MAX_LOSSTRADE	Pérdida máxima – el valor mínimo entre todas las transacciones de pérdidas. Su valor es inferior o igual a cero	double
STAT_CONPROFITMAX	Beneficio máximo en una secuencia de transacciones rentables. Su valor es superior o igual a cero	double
STAT_CONPROFITMAX_TRADES	Número de transacciones que han formado STAT_CONPROFITMAX (beneficio máximo en una secuencia de transacciones rentables)	int
STAT_MAX_CONWINS	Beneficio total en la serie más larga de transacciones rentables	double
STAT_MAX_CONPROFIT_TRADES	Numero de transacciones en la serie más larga de transacciones rentables STAT_MAX_CONWINS	int
STAT_CONLOSSMAX	Pérdida máxima en una secuencia de transacciones de pérdidas. Su valor es inferior o igual a cero	double
STAT_CONLOSSMAX_TRADES	Número de transacciones que han formado STAT_CONLOSSMAX (pérdida máxima en una secuencia de transacciones de pérdidas)	int
STAT_MAX_CONLOSSES	Pérdidas totales en la serie más larga de	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción del parámetro estadístico	Tipo
	transacciones de pérdidas	
STAT_MAX_CONLOSS_TRADES	Número de transacciones en la serie más larga de transacciones de pérdidas STAT_MAX_CONLOSSES	int
STAT_BALANCEMIN	Valor mínimo del balance	double
STAT_BALANCE_DD	Reducción máxima del balance en dinero. En el proceso de trading el balance puede sufrir varias reducciones, se coge el valor máximo.	double
STAT_BALANCEDD_PERCENT	Reducción del balance en por cientos que ha sido detectada en el momento de la reducción máxima del balance en dinero (STAT_BALANCE_DD).	double
STAT_BALANCE_DDREL_PERCENT	Reducción máxima del balance en por cientos. En el proceso de trading el balance puede sufrir varias reducciones, para cada una de ellas se calcula el valor relativo de reducción en por cientos. Se devuelve el valor máximo	double
STAT_BALANCE_DD_RELATIVE	Reducción del balance en dinero que ha sido detectada en el momento de la reducción máxima del balance en por cientos (STAT_BALANCE_DDREL_PERCENT).	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción del parámetro estadístico	Tipo
STAT_EQUITYMIN	Valor mínimo de equidad	double
STAT_EQUITY_DD	Reducción máxima de equidad en dinero. En el proceso de trading la equidad puede sufrir varias reducciones, se coge el valor máximo.	double
STAT_EQUITYDD_PERCENT	Reducción de equidad en por cientos que ha sido detectada en el momento de la reducción máxima de equidad en dinero (STAT_EQUITY_DD).	double
STAT_EQUITY_DDREL_PERCENT	Reducción máxima de equidad en por cientos. En el proceso de trading la equidad puede sufrir varias reducciones, para cada una de ellas se calcula el valor relativo de reducción en por cientos. Se devuelve el valor máximo	double
STAT_EQUITY_DD_RELATIVE	Reducción de equidad en dinero que ha sido detectada en el momento de la reducción máxima de equidad en por cientos (STAT_EQUITY_DDREL_PERCENT).	double
STAT_EXPECTED_PAYOFF	Beneficio esperado	double
STAT_PROFIT_FACTOR	Factor de beneficio - relación entre STAT_GROSS_PROFIT /STAT_GROSS_LOSS. Si STAT_GROSS_LOSS=0 , entonces el Factor	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción del parámetro estadístico	Tipo
	de Beneficio obtiene el valor DBL_MAX	
STAT_RECOVERY_FACTOR	Factor de recuperación - relación entre STAT_PROFIT/STAT_BALANCE_DD	double
STAT_SHARPE_RATIO	Ratio de Sharpe	double
STAT_MIN_MARGINLEVEL	Valor mínimo alcanzado del nivel de margen	double
STAT_CUSTOM_ONTESTER	Valor del criterio de optimización de usuario, calculado y devuelto por la función OnTester()	double
STAT_DEALS	Número de transacciones realizadas	int
STAT_TRADES	Número de trades	int
STAT_PROFIT_TRADES	Trades rentables	int
STAT_LOSS_TRADES	Trades irrentables	int
STAT_SHORT_TRADES	Trades cortos	int
STAT_LONG_TRADES	Trades largos	int
STAT_PROFIT_SHORTTRADES	Trades cortos rentables	int
STAT_PROFIT_LONGTRADES	Trades largos rentables	int
STAT_PROFITTRADES_AVGCON	Longitud media de una serie rentable de trades	int
STAT_LOSSTRADES_AVGCON	Longitud media de una serie irrentable de trades	int
STAT_COMPLEX_CRITERION	Criterio de optimización complejo	

© 2000-2025, MetaQuotes Ltd.

Constantes comerciales

Varias constantes que se usan para programar las estrategias comerciales están divididas en los siguientes grupos:

- Información sobre datos históricos del instrumento – la obtención de la información general sobre el instrumento financiera;
- Propiedades de órdenes – obtención de información sobre las órdenes comerciales;
- Propiedades de posiciones – obtención de información sobre las posiciones actuales;
- Propiedades de transacciones – obtención de información sobre las transacciones realizadas;
- Tipos de operaciones comerciales – descripción de las operaciones comerciales disponibles;
- Tipos de transacciones comerciales - descripción de posibles tipos de transacciones comerciales;
- Tipos de órdenes en profundidad de mercado - separación de las órdenes según la dirección de operación comercial solicitada.

© 2000-2025, MetaQuotes Ltd.

Información sobre datos históricos del instrumento

Cuando se accede a las series temporales se usa la función SeriesInfoInteger() para obtener la información adicional sobre el instrumento. El identificador de la propiedad requerida se pasa como el parámetro de esta función. Este identificador puede adquirir uno de los valores de la enumeración ENUM_SERIES_INFO_INTEGER.

ENUM_SERIES_INFO_INTEGER

Identificador	Descripción	Tipo de la propiedad
SERIES_BARS_COUNT	Número de barras para el símbolo-período en el momento actual	long
SERIES_FIRSTDATE	La primera fecha para el símbolo-período en el momento actual	datetime
SERIES_LASTBAR_DATE	Información sobre datos históricos del instrumento	datetime
SERIES_SERVER_FIRSTDATE	La primera fecha en el historial para el símbolo en el servidor independientemente del período	datetime
SERIES_TERMINAL_FIRSTDATE	La primera fecha en el historial para el símbolo en el terminal de cliente independientemente del período	datetime
SERIES_SYNCHRONIZED	Significa la sincronización de datos para el símbolo/período en este momento	bool

© 2000-2025, MetaQuotes Ltd.

Propiedades de órdenes

Ordenaciones de ejecutar las operaciones comerciales se formalizan mediante las órdenes. Cada orden posee una multitud de propiedades para la lectura. La información acerca de ellas se obtiene a través de la función OrderGet...() y HistoryOrderGet...().

Para las funciones OrderGetInteger() y HistoryOrderGetInteger()

ENUM_ORDER_PROPERTY_INTEGER

Identificador	Descripción	Tipo
ORDER_TICKET	Ticket de la orden. Un número único que se asigna a cada orden	long
ORDER_TIME_SETUP	Hora de establecimiento de la orden	datetime
ORDER_TYPE	Tipo de la orden	ENUM_ORDER_TYPE
ORDER_STATE	Estatus de la orden	ENUM_ORDER_STATE
ORDER_TIME_EXPIRATION	Plazo de expiración de la orden	datetime
ORDER_TIME_DONE	Hora de ejecución o cancelación de la orden	datetime
ORDER_TIME_SETUP_MSC	Tiempo de colocación de la orden para la ejecución en milisegundos desde 01.01.1970	long
ORDER_TIME_DONE_MSC	Tiempo de ejecución / retirada de la orden en milisegundos desde 01.01.1970	long

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo
ORDER_TYPE_FILLING	Tipo de ejecución según el resto	ENUM_ORDER_TYPE_FILLING
ORDER_TYPE_TIME	Tiempo de vida de la orden	ENUM_ORDER_TYPE_TIME
ORDER_MAGIC	Identificador del Asesor Experto que ha colocado la orden (sirve para que cada Asesor Experto ponga su único número personal)	long
ORDER_REASON	Motivo u origen de la colocación de una orden	ENUM_ORDER_REASON
ORDER_POSITION_ID	Identificador de posición que se coloca en la orden a la hora de su ejecución. Cada orden ejecutada provoca una transacción que abre una nueva posición, o cambia una que ya existe. El identificador de esta misma posición se establece para la orden ejecutada en este momento.	long

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo
ORDER_POSITION_BY_ID	Identificador de la posición opuesta para las órdenes del tipo ORDER_TYPE_CLOSE_BY.	long

Para las funciones OrderGetDouble() y HistoryOrderGetDouble()

ENUM_ORDER_PROPERTY_DOUBLE

Identificador	Descripción	Tipo
ORDER_VOLUME_INITIAL	Volumen inicial de la orden	double
ORDER_VOLUME_CURRENT	Volumen actual de la orden	double
ORDER_PRICE_OPEN	Precio especificado en la orden	double
ORDER_SL	Nivel Stop Loss	double
ORDER_TP	Nivel Take Profit	double
ORDER_PRICE_CURRENT	Precio actual del símbolo de la orden	double
ORDER_PRICE_STOPLIMIT	Precio Limit de la orden al activarse StopLimit	double

Para las funciones OrderGetString() y HistoryOrderGetString()

ENUM_ORDER_PROPERTY_STRING

Identificador	Descripción	Tipo
ORDER_SYMBOL	Símbolo de la orden	string
ORDER_COMMENT	Comentario	string

© 2000-2025, MetaQuotes Ltd.

Cuando se envía una solicitud comercial mediante la función OrderSend(), algunas operaciones requieren la indicación del tipo de la orden. El tipo de la orden se introduce en el campo *type* de una estructura especial MqlTradeRequest y puede adquirir los valores de la enumeración ENUM_ORDER_TYPE.

ENUM_ORDER_TYPE

Identificador	Descripción
ORDER_TYPE_BUY	Orden de mercado para la compra
ORDER_TYPE_SELL	Orden de mercado para la venta
ORDER_TYPE_BUY_LIMIT	Orden pendiente Buy Limit
ORDER_TYPE_SELL_LIMIT	Orden pendiente Sell Limit
ORDER_TYPE_BUY_STOP	Orden pendiente Buy Stop
ORDER_TYPE_SELL_STOP	Orden pendiente Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	Al alcanzar el precio de la orden se coloca la orden pendiente Buy Limit por el precio StopLimit
ORDER_TYPE_SELL_STOP_LIMIT	Al alcanzar el precio de la orden se coloca la orden pendiente Sell Limit por el precio StopLimit
ORDER_TYPE_CLOSE_BY	Orden de cierre de una posición con la opuesta

Cada orden tiene su estatus que describe su estado. Para obtener más detalles utilice la función OrderGetInteger() o HistoryOrderGetInteger() con el modificador ORDER_STATE. Los valores admisibles se almacenan en la enumeración ENUM_ORDER_STATE.

ENUM_ORDER_STATE

Identificador	Descripción
ORDER_STATE_STARTED	Orden verificada pero aún sin aceptar por el corredor
ORDER_STATE_PLACED	Orden aceptada
ORDER_STATE_CANCELED	Orden retirada por el cliente
ORDER_STATE_PARTIAL	Orden ejecutada parcialmente
ORDER_STATE_FILLED	Orden ejecutada totalmente
ORDER_STATE_REJECTED	Orden rechazada
ORDER_STATE_EXPIRED	Orden retirada por expirarse el plazo
ORDER_STATE_REQUEST_ADD	Orden en el estado de registro (colocación en el sistema de trading)
ORDER_STATE_REQUEST_MODIFY	Orden en el estado de modificación (cambio de parámetros)

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
ORDER_STATE_REQUEST_CANCEL	Orden en el estado de eliminación (eliminación del sistema de trading)

Al enviar una solicitud comercial de ejecución **en el momento actual** (time in force) se deberá indicar el precio y el volumen necesario de compra/venta. Hay que considerar que no existe garantía en los mercados financieros de que en un momento dado esté disponible para este instrumento financiero todo el volumen solicitado al precio deseado. Por consiguiente, el comercio en tiempo real está regulado por los modos de ejecución de precio y volumen. Los modos o políticas de ejecución determinan las reglas para los casos en los que ha cambiado el precio, o el volumen solicitado no se puede ejecutar al completo **en el momento actual**.

Modo de ejecución por precio: es posible obtenerlo de la propiedad del símbolo SYMBOL_TRADE_EXEMODE, que contiene una combinación de banderas de la enumeración ENUM_SYMBOL_TRADE_EXECUTION.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
Modo de ejecución (Request Execution)	Ejecución de una orden de mercado al precio recibido previamente del bróker. Antes de enviar una orden de mercado, se le solicitarán al bróker sus precios de ejecuci	SYMBOL_TRADE_EXECUTION_REQUEST

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	ón. Después de recibirlos, la ejecución de la orden a un precio determinado podrá confirmarse o rechazarse.	
Ejecución instantánea (Instant Execution)	Ejecución instantánea de una orden de mercado al precio indicado. Al enviar una solicitud comercial para su ejecución, la plataforma sustituirá automáticame	SYMBOL_TRADE_EXECUTION_INSTANT

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	nte los precios actuales en la orden. • Si el bróker acepta el precio, la orden se ejecutar	

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	á. • Si el bróker no acepta el precio solicitado, se produci	

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	rálallamada" recotización" (Requote): elbrókerretor	

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	narálosprecioslosquesepuedeejecutarestaorden.	

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
Ejecución por mercado (Market Execution)	El bróker tomará la decisión sobre el precio de ejecución sin un acuerdo adicional con el tráder. Enviar una orden de mercado en este modo implica la aceptación anticipada del precio al que se ejecutará.	SYMBOL_TRADE_EXECUTION_MARKET
Ejecución por bolsa (Exchange Execution)	Las operaciones comerciales se realizan según los precios	SYMBOL_TRADE_EXECUTION_EXCHANGE

© 2000-2025, MetaQuotes Ltd.

Modo de ejecución	Descripción	Valor en ENUM_SYMBOL_TRADE_EXECUTION
	de las ofertas de mercado actuales.	

Política de rellenado del volumen : se indica en la propiedad de la orden ORDER_TYPE_FILLING, y puede contener solo valores de la enumeración ENUM_ORDER_TYPE_FILLING

Política de ejecución	Descripción	Valor en ENUM_ORDER_TYPE_FILLING
Todo/Nada (Fill or Kill)	La orden solo puede ser ejecutada en el volumen indicado. Si en el mercado en estos momentos no existe volumen suficiente de un instrumento financiero, la orden no será	ORDER_FILLING_FOK

© 2000-2025, MetaQuotes Ltd.

Política de ejecución	Descripción	Valor en ENUM_ORDER_TYPE_FILLING
	ejecutada. El volumen necesario puede componerse de varias ofertas disponibles en el mercado en el momento actual. La posibilidad de usar órdenes FOK se determina en el servidor comercial.	
Todo/Parte (Immediate or Cancel)	El tráder acepta realizar una transacción al volumen	ORDER_FILLING_IOC

© 2000-2025, MetaQuotes Ltd.

Política de ejecución	Descripción	Valor en ENUM_ORDER_TYPE_FILLING
	máximo disponible en el mercado dentro del indicado en la orden. En caso de que no sea posible la ejecución completa, la orden se ejecutará con el volumen disponible, mientras que el volumen de la orden no ejecutado será cancelado. La posibilidad de	

© 2000-2025, MetaQuotes Ltd.

Política de ejecución	Descripción	Valor en ENUM_ORDER_TYPE_FILLING
	usar órdenes IOC se determina en el servidor comercial.	
Pasiva (Book or Cancel)	Una orden BoC implica que la orden solo puede colocarse en la Profundidad de Mercado y no puede ejecutarse inmediatamente. Si la orden puede ejecutarse inmediatamente al colocarse, entonces se	ORDER_FILLING_BOC

© 2000-2025, MetaQuotes Ltd.

Política de ejecución	Descripción	Valor en ENUM_ORDER_TYPE_FILLING
	eliminará. De hecho, la política de BoC garantiza que el precio de la orden colocada será peor que el del mercado actual. Las órdenes BoC se utilizan para implementar la negociación pasiva, de forma que se garantice que la orden no se ejecutará inmediatame	

© 2000-2025, MetaQuotes Ltd.

Política de ejecución	Descripción	Valor en ENUM_ORDER_TYPE_FILLING
	nte tras su colocación y no afecte a la liquidez actual. Solo se admite para órdenes límite y stop (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT, ORDER_TYPE_SELL_STOP_LIMIT).	
Devolver (Return)	En caso de ejecución parcial, la orden con el volume	ORDER_FILLING_RETURN

© 2000-2025, MetaQuotes Ltd.

Política de ejecución	Descripción	Valor en ENUM_ORDER_TYPE_FILLING
	n restante no se cancela, sino que continúa activa. Las órdenes Return no están permitidas en el modo de ejecución Market Execution (ejecución de órdenes por mercado — SYMBOL_TRADE_EXECUTION_MARKET).	

Al enviar una solicitud comercial usando la funciónOrderSend(), la política de ejecución necesaria por volumen se puede configurar en el campo *type_filling*, en la estructura especial MqlTradeRequest se permiten valores de la enumeración ENUM_ORDER_TYPE_FILLING. Para obtener el valor de esta propiedad en una orden activa/activada específica, se usará la función OrderGetInteger() o HistoryOrderGetInteger() con el modificador ORDER_TYPE_FILLING.

Antes de enviar una orden con ejecución en el momento actual, para establecer correctamente el valor ORDER_TYPE_FILLING (tipo de ejecución por volumen), podemos obtener para cada instrumento financiero con la ayuda de la función SymbolInfoInteger() el valor de la propiedad SYMBOL_FILLING_MODE, que muestra en forma de combinación de banderas los tipos de ejecución por

© 2000-2025, MetaQuotes Ltd.

volumen permitidos para este símbolo. Cabe señalar que el tipo de rellenado ORDER_FILLING_RETURN siempre está permitido, salvo para el modo "Ejecución por mercado" (SYMBOL_TRADE_EXECUTION_MARKET).

El uso de los tipos de rellenado según el modo de ejecución se puede representar en forma de recuadro:

Modo de ejecución\Política de rellenado	Todo/Nada (FOK ORDER_FILLING_FOK)	Todo/Parte (IOC ORDER_FILLING_IOC)	Devolver (Return ORDER_FILLING_RETURN)
Ejecución instantánea (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (independientemente de los ajustes del símbolo)	+ (independientemente de los ajustes del símbolo)	+ (siempre)
Ejecución por solicitud SYMBOL_TRADE_EXECUTION_REQUEST	+ (independientemente de los ajustes del símbolo)	+ (independientemente de los ajustes del símbolo)	+ (siempre)
Ejecución por mercado SYMBOL_TRADE_EXECUTION_MARKET	+ (se establece en los ajustes del símbolo)	+ (se establece en los ajustes del símbolo)	- (prohibido independientemente de los ajustes del símbolo)
Ejecución por bolsa SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (se establece en los ajustes del símbolo)	+ (se establece en los ajustes del símbolo)	+ (siempre)

Para las órdenes pendientes, independientemente del modo de ejecución (SYMBOL_TRADE_EXEMODE), se debe usar el tipo de rellenado ORDER_FILLING_RETURN, ya que dichas órdenes no están pensadas para ejecutarse en el momento del envío. Al usar órdenes pendientes, el tráder aceptará de antemano que cuando se den las condiciones para una transacción en esta orden, el bróker usará el tipo de rellenado que sea compatible con la plataforma comercial dada.

Se puede determinar el plazo de vigencia de la orden en el campo *type_time* de la estructura especial MqlTradeRequest a la hora de enviar la solicitud comercial mediante OrderSend(). Se admiten los valores de la enumeración ENUM_ORDER_TYPE_TIME. Para obtener el valor de esta propiedad utilice la función OrderGetInteger() o HistoryOrderGetInteger() con el modificador ORDER_TYPE_TIME.

ENUM_ORDER_TYPE_TIME

Identificador	Descripción
ORDER_TIME_GTC	Orden estará en la cola hasta que se retire

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
ORDER_TIME_DAY	Orden estará vigente sólo en el transcurso del día comercial corriente
ORDER_TIME_SPECIFIED	Orden estará vigente hasta que se expire el plazo de vigencia
ORDER_TIME_SPECIFIED_DAY	Orden se mantiene activa hasta las 23:59:59 del día especificado. Si está hora no cae en ninguna sesión de trading, la expiración llega a la hora de trading más cercana.

En la propiedad ORDER_REASON se contiene el motivo de la colocación de una orden. Una orden puede colocarse con la ayuda de un programa MQL5 o desde una aplicación móvil, o bien como resultado del evento StopOut, etcétera. Los posibles valores de ORDER_REASON se describen en la enumeración ENUM_ORDER_REASON.

ENUM_ORDER_REASON

Identificador	Descripción
ORDER_REASON_CLIENT	La orden se ha colocado desde el terminal de escritorio
ORDER_REASON_MOBILE	La orden se ha colocado desde la aplicación móvil
ORDER_REASON_WEB	La orden se ha colocado desde la plataforma web
ORDER_REASON_EXPERT	La orden se ha colocado desde un programa MQL5 - un asesor o script
ORDER_REASON_SL	La orden se ha colocado como resultado de la activación de un Stop Loss
ORDER_REASON_TP	La orden se ha colocado como resultado de la activación de un Take Profit
ORDER_REASON_SO	La orden se ha colocado como resultado del evento Stop Out

© 2000-2025, MetaQuotes Ltd.

Propiedades de posiciones

El resultado de ejecución de las operaciones comerciales es la apertura de una posición, cambio de su volumen y/o dirección, o su desaparición. Las operaciones comerciales se llevan a cabo a base de las órdenes enviadas por la función OrderSend() en forma de las solicitudes comerciales. Para cada instrumento financiero (símbolo) se prevé posible sólo una posición abierta. Una posición tiene un conjunto de propiedades disponibles para la lectura para las funciones PositionGet...().

Para la función PositionGetInteger()

ENUM_POSITION_PROPERTY_INTEGER

Identificador	Descripción	Tipo
POSITION_TICKET	Ticket de la posición. Un número único que se asigna a cada posición abierta de nuevo. Normalmente, corresponde al ticket de la orden como resultado de la cual se abrió la posición, excepto en los casos en los que el ticket ha cambiado debido a las operaciones de servicio en el servidor. Por ejemplo, el aumento de los swaps por la reapertura de posición. Para encontrar la orden con la que se ha abierto la posición, se debe usar la propiedad POSITION_IDENTIFIER. El valor POSITION_TICKET corresponde a	long

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo
	MqlTradeRequest::position.	
POSITION_TIME	Hora de apertura de posición	datetime
POSITION_TIME_MSC	Tiempo de apertura de la posición en milisegundos desde 01.01.1970	long
POSITION_TIME_UPDATE	Tiempo de modificación de la posición	datetime
POSITION_TIME_UPDATE_MSC	Tiempo de modificación de la posición en milisegundos desde 01.01.1970	long
POSITION_TYPE	Tipo de posición	ENUM_POSITION_TYPE
POSITION_MAGIC	Magic number para la posición (véase ORDER_MAGIC)	long
POSITION_IDENTIFIER	Identificador de posición es un número único que se adjudica a cada una de las posiciones abiertas y no se cambia a lo largo de su existencia. La rotación de posición no cambia su identificador. El identificador de la posición se indica en cada orden (ORDER_POSITION_ID) y operación	long

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo
	(DEAL_POSITION_ID) utilizada para abrirla, modificarla o cerrarla. Use esta propiedad para buscar órdenes u operaciones relacionadas con la posición. Durante el viraje de una posición en el modo de compensación (usando una única transacción de salida/entrada) el identificador de posición POSITION_IDENTIFIER no cambia. Sin embargo, el POSITION_TICKET es reemplazado por el ticket de la orden que ha conducido al viraje. En el modo de cobertura no se contempla el viraje de posición.	
POSITION_REASON	Causa de la apertura de una posición:	ENUM_POSITION_REASON

Para la función PositionGetDouble()

ENUM_POSITION_PROPERTY_DOUBLE

Identificador	Descripción	Tipo
POSITION_VOLUME	Volumen de posición	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo
POSITION_PRICE_OPEN	Precio de posición	double
POSITION_SL	Nivel Stop Loss para una posición abierta	double
POSITION_TP	Nivel Take Profit para una posición abierta	double
POSITION_PRICE_CURRENT	Precio actual para el símbolo	double
POSITION_SWAP	Swap acumulado	double
POSITION_PROFIT	Beneficio corriente	double

Para la función PositionGetString()

ENUM_POSITION_PROPERTY_STRING

Identificador	Descripción	Tipo
POSITION_SYMBOL	Símbolo de posición abierta	string
POSITION_COMMENT	Comentarios de posición	string
POSITION_EXTERNAL_ID	Identificador de la posición en el sistema externo de comercio (en la bolsa)	stringstring

La dirección de una posición abierta (compra o venta) se determina con los valores de la enumeración ENUM_POSITION_TYPE. Para obtener el tipo de una posición abierta, utilice la función PositionGetInteger() con el modificador POSITION_TYPE.

ENUM_POSITION_TYPE

Identificador	Descripción
POSITION_TYPE_BUY	Compra
POSITION_TYPE_SELL	Venta

© 2000-2025, MetaQuotes Ltd.

En la propiedad POSITION_REASON se contiene el motivo de la apertura de una posición. La posición puede abrirse como resultado de la activación de una orden colocada desde el terminal de escritorio, desde la aplicación móvil, con la ayuda de un asesor, etcétera. Los posibles valores de POSITION_REASON se describen en la enumeración ENUM_POSITION_REASON.

ENUM_POSITION_REASON

Identificador	Descripción
POSITION_REASON_CLIENT	La posición se ha abierto como resultado de la activación de una orden colocada desde el terminal de escritorio
POSITION_REASON_MOBILE	La posición se ha abierto como resultado de la activación de una orden colocada desde la aplicación móvil
POSITION_REASON_WEB	La posición se ha abierto como resultado de la activación de una orden colocada desde la plataforma web
POSITION_REASON_EXPERT	La posición se ha abierto como resultado de la activación de una orden colocada desde un programa MQL5 - un asesor o script

© 2000-2025, MetaQuotes Ltd.

Propiedades de transacciones

Una transacción refleja el hecho de ejecución de una operación comercial a base de una orden que contiene una disposición comercial. Cada transacción se describe por las propiedades que permiten obtener información sobre ella. Para leer los valores de las propiedades se utilizan las funciones del tipo HistoryDealGet...() que devuelven los valores de las enumeraciones correspondientes.

Para la función HistoryDealGetInteger()

ENUM_DEAL_PROPERTY_INTEGER

Identificador	Descripción	Tipo
DEAL_TICKET	Ticket de la operación. Número único que se asigna a cada operación	long
DEAL_ORDER	Orden a base de la cual la transacción ha sido ejecutada	long
DEAL_TIME	Hora de ejecución de transacción	datetime
DEAL_TIME_MSC	Tiempo de ejecución de la transacción en milisegundos desde 01.01.1970	long
DEAL_TYPE	Tipo de transacción	ENUM_DEAL_TYPE
DEAL_ENTRY	Dirección de transacción - entra en el mercado, sale del mercado o da la vuelta	ENUM_DEAL_ENTRY
DEAL_MAGIC	Magic number para la transacción (véase ORDER_MAGIC)	long
DEAL_REASON	Causa u origen de la realización de la transacción	ENUM_DEAL_REASON
DEAL_POSITION_ID	Identificador de posición, en la apertura, modificación o cierre de la cual participa esta transacción. Cada posición tiene su único identificador que se asigna a todas las transacciones realizadas con el instrumento durante toda la vida de la posición.	long

Para la función HistoryDealGetDouble()

ENUM_DEAL_PROPERTY_DOUBLE

Identificador	Descripción	Tipo
DEAL_VOLUME	Volumen de transacción	double
DEAL_PRICE	Precio de transacción	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo
DEAL_COMMISSION	Comisión de transacción	double
DEAL_SWAP	Swap acumulado con el cierre	double
DEAL_PROFIT	Beneficio de transacción	double
DEAL_FEE	El pago por la realización de una transacción se efectúa justo después de ejecutar la misma.	double
DEAL_SL	Nivel de Stop Loss • Para una operación de entrada o vuelta, el valor de Stop Loss se toma de la orden que abre o da vuelta a la posición. • Para una operación de salida, el valor de Stop Loss se toma del cierre de la posició	double
DEAL_TP	Nivel de Nível Take Profit • Para una operación de entrada o vuelta, el valor de Take Profit se toma de la orden que abre o da vuela a la posición • Para una operación de salida, el valor de Take Profit	double

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo
	se toma del cierre de la posición	

Para la función HistoryDealGetString()

ENUM_DEAL_PROPERTY_STRING

Identificador	Descripción	Tipo
DEAL_SYMBOL	Símbolo de transacción	string
DEAL_COMMENT	Comentarios de transacción	string

Cada transacción se caracteriza por el tipo; los posibles valores se encuentran en la enumeración ENUM_DEAL_TYPE. Para obtener la información sobre el tipo de la transacción, utilice la función HistoryDealGetInteger() con el modificador DEAL_TYPE.

ENUM_DEAL_TYPE

Identificador	Descripción
DEAL_TYPE_BUY	Compra
DEAL_TYPE_SELL	Venta
DEAL_TYPE_BALANCE	Balance
DEAL_TYPE_CREDIT	Crédito
DEAL_TYPE_CHARGE	Cargas adicionales
DEAL_TYPE_CORRECTION	Corrección
DEAL_TYPE_BONUS	Bonos
DEAL_TYPE_COMMISSION	Comisiones adicionales
DEAL_TYPE_COMMISSION_DAILY	Comisión calculada al final de la jornada de trading
DEAL_TYPE_COMMISSION_MONTHLY	Comisión calculada al final del mes
DEAL_TYPE_COMMISSION_AGENT_DAILY	Comisión de agente calculada al final de la jornada de trading
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Comisión de agente calculada al final del mes
DEAL_TYPE_INTEREST	Calculo de intereses sobre fondos libres

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
DEAL_TYPE_BUY_CANCELED	Transacción de compra cancelada. Puede surgir la situación cuando una transacción de compra realizada anteriormente se cancele. En este caso el tipo de la operación realizada (DEAL_TYPE_BUY) se cambia a DEAL_TYPE_BUY_CANCELED, y su beneficio/pérdida se anula. El beneficio/pérdida producido/a anteriormente se carga/se quita de la cuenta por medio de una operación contable separada
DEAL_TYPE_SELL_CANCELED	Transacción de venta cancelada. Puede surgir la situación cuando una transacción de venta realizada anteriormente se cancele. En este caso el tipo de la operación realizada (DEAL_TYPE_SELL) se cambia a DEAL_TYPE_SELL_CANCELED, y su beneficio/pérdida se anula. El beneficio/pérdida producido/a anteriormente se carga/se quita de la cuenta por medio de una operación contable separada
DEAL_DIVIDEND	Operaciones con dividendos
DEAL_DIVIDEND_FRANKED	Operaciones con dividendos franqueados (no tributables)
DEAL_TAX	Carga impositiva

Las transacciones se diferencian no sólo por sus tipos que se establecen de la enumeración ENUM_DEAL_TYPE, sino por el modo de cambiar la posición. Esto puede ser una simple apertura de posición, o acumulación de una posición abierta anteriormente (entrada en el mercado), cierre de posición con una transacción de dirección opuesta de un volumen correspondiente (salida del mercado), o bien, la vuelta de posición en caso cuando el volumen de transacción en dirección opuesta supera el volumen de la posición abierta anteriormente.

Todas estas situaciones se describen por los valores de la enumeración ENUM_DEAL_ENTRY. para obtener esta información, utilice la función HistoryDealGetInteger() con el modificador DEAL_ENTRY.

ENUM_DEAL_ENTRY

Identificador	Descripción
DEAL_ENTRY_IN	Entrada en el mercado
DEAL_ENTRY_OUT	Salida del mercado
DEAL_ENTRY_INOUT	Vuelta
DEAL_ENTRY_OUT_BY	Cierre con la posición opuesta

En la propiedad DEAL_REASON se contiene el motivo de la realización de la transacción. La transacción puede realizarse como resultado de la activación de una orden colocada desde la aplicación móvil o desde un programa MQL5; o bien como resultado del evento StopOut o el abono/retirada del margen de variación, etcétera. Los posibles valores de DEAL_REASON se describen en la enumeración

© 2000-2025, MetaQuotes Ltd.

ENUM_DEAL_REASON. Para algunas transacciones provocadas por operaciones de cambio de balance, crédito, abono de comisiones, etcétera, se indica como causa DEAL_REASON_CLIENT.

ENUM_DEAL_REASON

Identificador	Descripción
DEAL_REASON_CLIENT	La transacción se realizado como resultado de la activación de una orden colocada desde el terminal de escritorio
DEAL_REASON_MOBILE	La transacción se realizado como resultado de la activación de una orden colocada desde la aplicación móvil
DEAL_REASON_WEB	La transacción se realizado como resultado de la activación de una orden colocada desde la plataforma web
DEAL_REASON_EXPERT	La transacción se realizado como resultado de la activación de una orden colocada desde un programa MQL5 - un asesor o script
DEAL_REASON_SL	La transacción se realizado como resultado de la activación de una orden Stop Loss
DEAL_REASON_TP	La transacción se realizado como resultado de la activación de una orden Take Profit
DEAL_REASON_SO	La transacción se realizado como resultado del evento Stop Out
DEAL_REASON_ROLLOVER	La transacción se realizado a causa del traslado de una posición
DEAL_REASON_VMARGIN	La transacción se realizado después de abonarse/retirarse el margen de variación
DEAL_REASON_SPLIT	La transacción se realizado a causa del fraccionamiento (reducción del precio) de un instrumento que tenía una posición abierta en el momento del fraccionamiento
DEAL_REASON_CORPORATE_ACTION	La transacción se ha realizado como resultado de una acción corporativa: fusión o cambio del nombre de un valor, transferencia de un cliente a otra cuenta, etc.

© 2000-2025, MetaQuotes Ltd.

Tipos de transacciones comerciales

La actividad comercial se realiza mediante el envío de las ordenaciones de apertura de posiciones usando la función OrderSend(), también mediante las ordenaciones de colocación, modificación y eliminación de órdenes pendientes. Cada orden comercial contiene la especificación del tipo de la operación comercial solicitada. Las operaciones comerciales se describen en la enumeración ENUM_TRADE_REQUEST_ACTIONS.

ENUM_TRADE_REQUEST_ACTIONS

Identificador	Descripción
TRADE_ACTION_DEAL	Colocar una orden comercial de conclusión inmediata de un transacción según los parámetros especificados (colocar una orden de mercado)
TRADE_ACTION_PENDING	Colocar una orden comercial para la conclusión de una transacción bajo unas condiciones especificadas (orden pendiente)
TRADE_ACTION_SLTP	Modificar los valores Stop Loss y Take Profit de una posición abierta
TRADE_ACTION_MODIFY	Modificar los parámetros de una orden comercial colocada anteriormente
TRADE_ACTION_REMOVE	Eliminar una orden pendiente colocada anteriormente
TRADE_ACTION_CLOSE_BY	Close a position by an opposite one

Ejemplo de la operación comercial TRADE_ACTION_DEAL para abrir una posición Buy:

```
#define EXPERT_MAGIC 123456   // Número mágico del experto
//+------------------------------------------------------------------+
//| Apertura de posición Buy                                         |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración e inicialización de la solicitud y el resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parámetros de la solicitud
   request.action   =TRADE_ACTION_DEAL;                  // tipo de operación come
   request.symbol   =Symbol();                           // símbolo
   request.volume   =0.1;                                // volumen de 0.1 lote
   request.type     =ORDER_TYPE_BUY;                     // tipo de orden
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // precio de apertura
   request.deviation=5;                                  // desviación permisible
   request.magic    =EXPERT_MAGIC;                       // Número mágico de la or
//--- envío de la solicitud
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // si no se ha logrado en
//--- información sobre la operación
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

Ejemplo de la operación comercial TRADE_ACTION_DEAL para abrir una posición Sell:

```
#define EXPERT_MAGIC 123456   // Número mágico del experto
//+------------------------------------------------------------------+
//| Apertura de posición Sell                                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración e inicialización de la solicitud y el resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parámetros de la solicitud
   request.action   =TRADE_ACTION_DEAL;                     // tipo de operación comer
   request.symbol   =Symbol();                              // símbolo
   request.volume   =0.2;                                   // volumen 0.2 lote
   request.type     =ORDER_TYPE_SELL;                       // tipo de orden
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // precio de apertura
   request.deviation=5;                                     // desviación permisible d
   request.magic    =EXPERT_MAGIC;                          // Número mágico de la orc
//--- envío de la solicitud
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // si no se ha logrado env
//--- información sobre la operación
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Ejemplo de la operación comercial TRADE_ACTION_DEAL para cerrar una posición:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // Número mágico del experto
//+------------------------------------------------------------------+
//| Cierre de todas las posiciones                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración de la solicitud y el resultado
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // número de posiciones abiertas
//--- iteración de todas las posiciones abiertas
   for(int i=total-1; i>=0; i--)
     {
      //--- parámetros de la orden
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- mostrar la información de la posición
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- si el número mágico coincide
      if(magic==EXPERT_MAGIC)
        {
         //--- reseteo de los valores de la solicitud y el resultado
         ZeroMemory(request);
         ZeroMemory(result);
         //--- establecer los parámetros de la operación
         request.action   =TRADE_ACTION_DEAL;        // tipo de operación comercial
         request.position =position_ticket;          // ticket de la posición
         request.symbol   =position_symbol;          // símbolo
         request.volume   =volume;                   // volumen de la posición
         request.deviation=5;                        // desviación permisible del prec
         request.magic    =EXPERT_MAGIC;             // Número mágico de la posición
         //--- establecer el precio y el tipo de orden dependiendo del tipo de posici
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
           }
         //--- mostrar información sobre el cierre
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- envío de la solicitud
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // si no se ha logrado
         //--- información sobre la operación
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
//+------------------------------------------------------------------+
```

Ejemplo de la operación comercial TRADE_ACTION_PENDING para colocar una orden pendiente:

© 2000-2025, MetaQuotes Ltd.

```
#property description "Ejemplo de colocación de órdenes pendientes"
#property script_show_inputs
#define EXPERT_MAGIC 123456                              // Número mágico del experto
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;    // tipo de orden
//+------------------------------------------------------------------+
//| Colocar órdenes pendientes                                       |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración e inicialización de la solicitud y el resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parámetros para la colocación de una orden pendiente
   request.action   =TRADE_ACTION_PENDING;                                  // tipo de oper
   request.symbol   =Symbol();                                              // símbolo
   request.volume   =0.1;                                                   // volumen 0.1
   request.deviation=2;                                                     // desviación p
   request.magic    =EXPERT_MAGIC;                                          // Número mágic
   int offset = 50;                                                         // distancia co
   double price;                                                            // precio de ac
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);                     // tamaño del p
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);                     // número de dí
   //--- comprobación del tipo de operación
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                               // tipo de orde
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;             // precio de ap
      request.price    =NormalizeDouble(price,digits);                      // precio de ap
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                               // tipo de orc
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;             // precio de a
      request.price    =NormalizeDouble(price,digits);                      // precio de a
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                                    // tipo de orc
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;     // precio de a
      request.price=NormalizeDouble(price,digits);                          // precio de a
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                               // tipo de orc
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;             // precio de a
      request.price    =NormalizeDouble(price,digits);                      // precio de a
     }
   else Alert("Este ejemplo es solo para colocar órdenes pendientes");    // si no se h
//--- enviar solicitud
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());                     // si no se ha
//--- información sobre la operación
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Ejemplo de la operación comercial TRADE_ACTION_SLTP para cambiar los valores de Stop Loss y Take Profit de una posición abierta:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // Número mágico del experto
//+------------------------------------------------------------------+
//| Modificación de Stop Loss y Take Profit de la posición           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración de la solicitud y el resultado
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // número de posiciones abiertas
//--- iteración de todas las posiciones abiertas
   for(int i=0; i<total; i++)
     {
      //--- parámetros de la orden
      ulong  position_ticket=PositionGetTicket(i);// ticket de la posición
      string position_symbol=PositionGetString(POSITION_SYMBOL); // símbolo
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // número d
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // Número mágico de la posición
      double volume=PositionGetDouble(POSITION_VOLUME);    // volumen de la posición
      double sl=PositionGetDouble(POSITION_SL);  // Stop Loss de la posición
      double tp=PositionGetDouble(POSITION_TP);  // Take Profit de la posición
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- mostrar información de la posición
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- si el número mágico coincide, Stop Loss y Take Profit no han sido estable
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- cálculo de los niveles de precio actuales
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_L
         double price_level;
         //--- si el nivel de separación máxima permitida en puntos con respecto al pr
         if(stop_level<=0)
            stop_level=150; // establecemos una separación de 150 puntos con respecto
         else
            stop_level+=50; // tomaremos un nivel de separación igual a (SYMBOL_TRADE_

         //--- cálculo y redondeo de los valores de Stop Loss y Take Profit
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(ask+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(bid-price_level,digits);
           }
         //--- reseteo de los valores de la solicitud y el resultado
         ZeroMemory(request);
         ZeroMemory(result);
         //--- establecer los parámetros de la operación
         request.action  =TRADE_ACTION_SLTP; // tipo de operación comercial
         request.position=position_ticket;   // ticket de la posición
         request.symbol=position_symbol;     // símbolo
         request.sl      =sl;                // Stop Loss de la posición
         request.tp      =tp;                // Take Profit de la posición
         request.magic=EXPERT_MAGIC;         // Número mágico de la posición
         //--- mostrar información de la modificación
         PrintFormat("Modify #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- envío de la solicitud
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // si no se ha logrado
         //--- información sobre la operación
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
   }
//+------------------------------------------------------------------+
```

Ejemplo de la operación comercial TRADE_ACTION_MODIFY para modificar los niveles de precio de las órdenes pendientes:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // Número mágico del experto
//+------------------------------------------------------------------+
//| Modificación de órdenes pendientes                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración e incialización de la solicitud y el resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // número de órdenes pendientes colocadas
//--- iteración de todas las órdenes pendientes colocadas
   for(int i=0; i<total; i++)
     {
      //--- parámetros de la orden
      ulong  order_ticket=OrderGetTicket(i);                          // ticket de
      string order_symbol=Symbol();                                   // símbolo
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS); // número de
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                      // Número mág
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);             // volumen ac
      double sl=OrderGetDouble(ORDER_SL);                             // Stop Loss
      double tp=OrderGetDouble(ORDER_TP);                             // Take Profi
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // tipo de or
      int offset = 50;                                                // distancia
      double price;                                                   // precio de
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);       // tamaño del
      //--- mostrar información sobre la orden
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  order_ticket,
                  order_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- si el número mágico coincide, Stop Loss y Take Profit no han sido establec
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                          // tipo de oper
         request.order = OrderGetTicket(i);                           // ticket de la
         request.symbol   =Symbol();                                  // símbolo
         request.deviation=5;                                         // desviación p
        //--- establecer los niveles de precio, Take Profit y Stop Loss de la orden de
         if(type==ORDER_TYPE_BUY_LIMIT)
           {
            price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // precio
           }
         else if(type==ORDER_TYPE_SELL_LIMIT)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // preci
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
```

© 2000-2025, MetaQuotes Ltd.

```
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // precio
           }
         else if(type==ORDER_TYPE_SELL_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // precio
           }
         //--- envío de la solicitud
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // si no se ha logrado
         //--- información sobre la operación
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //--- reseteo de los valores de la solicitud y el resultado
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

Ejemplo de la operación comercial TRADE_ACTION_REMOVE para eliminar órdenes pendientes:

```
#define EXPERT_MAGIC 123456  // Número mágico del experto
//+------------------------------------------------------------------+
//| Eliminación de órdenes pendientes                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración e inicialización de la solicitud y el resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // cantidad de órdenes pendientes establecidas
//--- iteración de todas las órdenes pendientes establecidas
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // ticket de la orden
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // Número mágico de la
      //--- si el número mágico coincide
      if(magic==EXPERT_MAGIC)
        {
         //--- reseteo de los valores de la solicitud y el resultado
         ZeroMemory(request);
         ZeroMemory(result);
         //--- establecer los parámetros de la operación
         request.action=TRADE_ACTION_REMOVE;                   // tipo de operación co
         request.order = order_ticket;                         // ticket de la orden
         //--- envío de la solicitud
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // si no se ha logrado
         //--- información sobre la operación
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

Ejemplo de la operación comercial TRADE_ACTION_CLOSE_BY para cerrar una posición con otra opuesta:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // Número mágico del experto
//+------------------------------------------------------------------+
//| Cierre de todas las posiciones                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración de la solicitud y el resultado
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // número de posiciones abiertas
//--- iteración de todas las posiciones abiertas
   for(int i=total-1; i>=0; i--)
     {
      //--- parámetros de la orden
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      double sl=PositionGetDouble(POSITION_SL);
      double tp=PositionGetDouble(POSITION_TP);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- mostrar la información de la posición
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- si el número mágico coincide
      if(magic==EXPERT_MAGIC)
        {
         for(int j=0; j<i; j++)
           {
            string symbol=PositionGetSymbol(j); // símbolo de la posición opuesta
            //--- si los símbolos de la posición opuesta y la buscada coinciden
            if(symbol==position_symbol && PositionGetInteger(POSITION_MAGIC)==EXPERT_M
              {
               //--- establecer el tipo de posición opuesta
               ENUM_POSITION_TYPE type_by=(ENUM_POSITION_TYPE)PositionGetInteger(POSIT
               //--- salir si los tipos de la posición original y la opuesta coinciden
               if(type==type_by)
                  continue;
               //--- reseteo de los valores de la solicitud y el resultado
               ZeroMemory(request);
               ZeroMemory(result);
               //--- establecer los parámetros del resultado
               request.action=TRADE_ACTION_CLOSE_BY;                    // tipo d
               request.position=position_ticket;                        // ticket
               request.position_by=PositionGetInteger(POSITION_TICKET); // ticket
               //request.symbol     =position_symbol;
               request.magic=EXPERT_MAGIC;                              // Número
               //--- mostrar la información sobre el cierre por posición opuesta
               PrintFormat("Close #%I64d %s %s by #%I64d",position_ticket,position_sym
               //--- envío de la solicitud
               if(!OrderSend(request,result))
                  PrintFormat("OrderSend error %d",GetLastError()); // si no se ha log
```

© 2000-2025, MetaQuotes Ltd.

```
                //--- información sobre la operación
                PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result
               }
            }
         }
      }
   }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

Tipos de transacciones (transactions) comerciales

Como resultado de ejecución de ciertas acciones con la cuenta de trading su estado se cambia. A estas acciones les pertenecen:

- El envío de una solicitud comercial por parte de cualquier aplicación MQL5 en el terminal de cliente utilizando la función OrderSend y OrderSendAsync, con su posterior ejecución;
- El envío de una solicitud comercial a través de la interfaz gráfica del terminal y su posterior ejecución;
- El accionamiento de órdenes pendientes y órdenes Stop en el servidor;
- La ejecución de operaciones en el servidor de trading.

Como resultado de estas acciones, para la cuenta se ejecutan las transacciones comerciales:

- tramitación de la solicitud comercial;
- cambio de órdenes abiertas;
- cambio del historial de órdenes;
- cambio del historial de operaciones;
- cambio de posiciones.

Por ejemplo, al enviar una orden de compra, ésta se tramita, para la cuenta se crea una orden de compra correspondiente, se realiza la ejecución de la orden, su eliminación de la lista de las abiertas, se agrega al historial de órdenes, luego la operación correspondiente se agrega al historial, y se crea una posición nueva. Pues todas estas acciones son transacciones comerciales.

Para que el programador pueda monitorear las acciones realizadas respecto a la cuenta de trading, está prevista la función OnTradeTransaction. Este manejador permite recibir en la aplicación MQL5 las transacciones comerciales que han sido aplicadas a la cuenta. La descripción de una transacción comercial se envía en el primer parámetro de OnTradeTransaction a través de la estructura MqlTradeTransaction.

El tipo de la transacción comercial se envía en el parámetro type de la estructura MqlTradeTransaction. Los posibles tipos de transacciones comerciales se describen en la enumeración:

ENUM_TRADE_TRANSACTION_TYPE

Identificador	Descripción
TRADE_TRANSACTION_ORDER_ADD	Agregación de una nueva orden abierta.
TRADE_TRANSACTION_ORDER_UPDATE	Modificación de orden abierta. A estas modificaciones les corresponden no sólo los cambios explícitos por parte del terminal de cliente o servidor de trading, sino también alteraciones de su estado durante la colocación (por ejemplo, paso del estado ORDER_STATE_STARTED a ORDER_STATE_PLACED o de ORDER_STATE_PLACED a ORDER_STATE_PARTIAL etc.).
TRADE_TRANSACTION_ORDER_DELETE	Eliminación de la orden de la lista de órdenes abiertas. Una orden puede ser eliminada de las abiertas como resultado de colocación de la solicitud

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
	correspondiente, o bien una vez ejecutada (llena) y pasada al historial.
TRADE_TRANSACTION_DEAL_ADD	Agregación de una transacción (deal) al historial. Se hace como resultado de ejecución de la orden o realización de la operación con el balance de la cuenta.
TRADE_TRANSACTION_DEAL_UPDATE	Modificación de una transacción (deal) en el historial. Puede pasar que una transacción (deal) ejecutada antes se cambie en el servidor. Por ejemplo, la transacción (deal) fue modificada en el sistema externo de trading (bolsa) a donde había sido pasada por el broker.
TRADE_TRANSACTION_DEAL_DELETE	Eliminación de una transacción (deal) del historial. Puede pasar que una transacción (deal) ejecutada antes se elimine en el servidor. Por ejemplo, la transacción (deal) fue eliminada en el sistema externo de trading (bolsa) a donde había sido pasada por el broker.
TRADE_TRANSACTION_HISTORY_ADD	Agregación de una orden al historial como resultado de su ejecución o cancelación.
TRADE_TRANSACTION_HISTORY_UPDATE	Modificación de una orden que se encuentra en el historial de órdenes. Este tipo está previsto para ampliar la funcionalidad en la parte del servidor.
TRADE_TRANSACTION_HISTORY_DELETE	Eliminación de una orden del historial de órdenes. Este tipo está previsto para ampliar la funcionalidad en la parte del servidor.
TRADE_TRANSACTION_POSITION	Modificación de la posición que no está relacionada con la ejecución de la transacción (deal). Este tipo de transacción (transaction) quiere decir que la posición ha sido modificada en la parte del servidor de trading. La posición puede sufrir el cambio del volumen, precio de apertura, así como de los niveles Stop Loss y Take Profit. La información sobre los cambios se envía en la estructura MqlTradeTransaction usando el manejador OnTradeTransaction. La modificación de una posición (agregación, cambio o eliminación) como resultado de ejecución de la transacción (deal) no supone la aparición tras sí la transacción (transaction) TRADE_TRANSACTION_POSITION.
TRADE_TRANSACTION_REQUEST	Aviso de que la solicitud comercial ha sido procesada por el servidor, y el resultado de su procesamiento ha sido recibido. Para las transacciones de este tipo en la estructura MqlTradeTransaction hay que analizar

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
	sólo un campo - type (tipo de transacción). Para obtener la información adicional hay que analizar el segundo y el tercer parámetro de la función OnTradeTransaction (request y result).

En función del tipo de la transacción comercial, en la estructura MqlTradeTransaction que la describe, se rellenan diferentes parámetros. La descripción detallada de los datos traspasados se puede encontrar en el apartado "Estructura de transacción comercial".

Véase también

Estructura de transacción comercial, OnTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Tipos de órdenes en la profundidad de mercado

Para los instrumentos bursátiles está disponible la ventana "Profundidad de Mercado", donde se puede ver las actuales órdenes de compra y venta. Para cada orden se especifica la dirección de operación comercial deseada, volumen requerido y el precio solicitado.

Para recibir la información sobre el estado actual de la profundidad de mercado utilizando los medios del lenguaje MQL5, tenemos la función MarketBookGet(). Esta función coloca "screenshot de la profundidad de mercado" en la matriz de estructuras MqlBookInfo. Cada elemento de esta matriz contiene información en el campo *type* sobre la dirección de la orden, es el valor de la enumeración ENUM_BOOK_TYPE.

ENUM_BOOK_TYPE

Identificador	Descripción
BOOK_TYPE_SELL	Orden de venta
BOOK_TYPE_BUY	Orden de compra
BOOK_TYPE_SELL_MARKET	Solicitud de venta por el precio de mercado
BOOK_TYPE_BUY_MARKET	Solicitud de compra por el precio de mercado

Véase también

Estructuras y clases, Estructura de profundidad de mercado, Tipos de operaciones comerciales, Obtención de información de mercado

© 2000-2025, MetaQuotes Ltd.

Propiedades de las señales

Valores de las enumeraciones para trabajar con las señales comerciales y los ajustes de su copiado.

Enumeraciones de las propiedades del tipo double de las señales comerciales:

ENUM_SIGNAL_BASE_DOUBLE

Constante	Descripción
SIGNAL_BASE_BALANCE	Balance de la cuenta
SIGNAL_BASE_EQUITY	Fondos en la cuenta
SIGNAL_BASE_GAIN	Crecimiento de la cuenta en por cientos
SIGNAL_BASE_MAX_DRAWDOWN	Reducción máxima
SIGNAL_BASE_PRICE	Precio de la suscripción a la señal
SIGNAL_BASE_ROI	Valor ROI (Return on Investment) de la señal en %

Enumeraciones de las propiedades del tipo integer de las señales comerciales:

ENUM_SIGNAL_BASE_INTEGER

Constante	Descripción
SIGNAL_BASE_DATE_PUBLISHED	Fecha de publicación de la señal (cuando se ha hecho disponible)
SIGNAL_BASE_DATE_STARTED	Fecha del inicio del monitoreo de la señal
SIGNAL_BASE_DATE_UPDATED	Fecha de la última actualización de la estadística comercial de la señal
SIGNAL_BASE_ID	ID de la señal
SIGNAL_BASE_LEVERAGE	Apalancamiento de la cuenta de trading
SIGNAL_BASE_PIPS	Resultado del trading en pips
SIGNAL_BASE_RATING	Posición en el ranking de las señales
SIGNAL_BASE_SUBSCRIBERS	Número de suscriptores
SIGNAL_BASE_TRADES	Número de trades
SIGNAL_BASE_TRADE_MODE	Tipo de la cuenta (0-real, 1-demo, 2-de concurso)

Enumeraciones de las propiedades del tipo string de las señales comerciales:

ENUM_SIGNAL_BASE_STRING

Constante	Descripción
SIGNAL_BASE_AUTHOR_LOGIN	Login del autor de la señal

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción
SIGNAL_BASE_BROKER	Nombre del broker (empresa)
SIGNAL_BASE_BROKER_SERVER	Servidor del broker
SIGNAL_BASE_NAME	Nombre de la señal
SIGNAL_BASE_CURRENCY	Divisa de la cuenta de la señal

Enumeraciones de las propiedades del tipo double de los ajustes del copiado las señales comerciales:

ENUM_SIGNAL_INFO_DOUBLE

Constante	Descripción
SIGNAL_INFO_EQUITY_LIMIT	Porcentaje para la conversión del volumen de la transacción
SIGNAL_INFO_SLIPPAGE	Valor del deslizamiento con el que se coloca la orden de mercado durante la sincronización y copiado de las transacciones
SIGNAL_INFO_VOLUME_PERCENT	Valor de limitación de fondos para la señal, r/o

Enumeraciones de las propiedades del tipo integer de los ajustes del copiado las señales comerciales:

ENUM_SIGNAL_INFO_INTEGER

Constante	Descripción
SIGNAL_INFO_CONFIRMATIONS_DISABLED	Bandera del permiso para la sincronización sin mostrar el diálogo de confirmación
SIGNAL_INFO_COPY_SLTP	Bandera del copiado de Stop Loss y Take Profit
SIGNAL_INFO_DEPOSIT_PERCENT	Limitación del depósito (en %)
SIGNAL_INFO_ID	id de la señal, r/o
SIGNAL_INFO_SUBSCRIPTION_ENABLED	Bandera del permiso para el copiado de las transacciones según la suscripción
SIGNAL_INFO_TERMS_AGREE	Bandera de aceptación de las condiciones del uso del servicio "Señales", r/o

Enumeraciones de las propiedades del tipo string de los ajustes del copiado las señales comerciales:

ENUM_SIGNAL_INFO_STRING

Constante	Descripción
SIGNAL_INFO_NAME	Nombre de la señal, r/o

Véase también

Administrar señales

© 2000-2025, MetaQuotes Ltd.

Constantes nombradas

Todas las constantes utilizadas en el lenguaje MQL5 pueden ser divididas en siguientes grupos:

- Macro substituciones predefinidas - los valores se substituyen durante la compilación;
- Constantes matemáticas - los valores de algunas expresiones matemáticas;
- Constantes de tipos numéricos - restricciones aplicadas a algunos tipos simples;
- Razones de deinicialización - descripción de las razones de deinicialización;
- Verificación del puntero a objeto - enumeración de los tipos de punteros devueltos por la función CheckPointer() ;
- Otras constantes - todas las demás constantes.

© 2000-2025, MetaQuotes Ltd.

Macro substituciones predefinidas

Para facilitar la depuración y obtener la información sobre el funcionamiento de un programa mql5 están previstas las constantes-macros especiales predefinidas, cuyos valores se establecen en el momento de compilación. La manera más fácil de usarlas consiste en la devolución de sus valores mediante la función Print(), como se muestra en el ejemplo de abajo.

Constante	Descripción
__CPU_ARCHITECTURE__	Nombre de la arquitectura (conjunto de comandos) con la que se ha creado el archivo EX5 al realizar la compilación
__DATE__	Fecha de compilación del archivo sin hora (horas, minutos y segundos son iguales a 0)
__DATETIME__	Fecha y hora de compilación del archivo
__LINE__	Número de cadena en el código fuente, en la que se ubica esta macro
__FILE__	Nombre del archivo corriente compilado
__PATH__	Ruta absoluta hacia el archivo actual a compilar
__FUNCTION__	Nombre de la función, en cuyo cuerpo está ubicado la macro
__FUNCSIG__	Signatura de la función en cuyo cuerpo se encuentra la macro. El mostrar en el log la descripción completa de la función con tipos de parámetros puede ser útil durante la identificación de las funciones sobrecargadas
__MQLBUILD__, __MQL5BUILD__	Número build del compilador
__COUNTER__	Para cada declaración __COUNTER__ encontrada, el compilador cambia en su lugar el valor del contador de 0 a N-1, donde N es el número de usos en el código. Al recompilar el código fuente sin cambios, el orden __COUNTER__ está garantizado. El valor del contador __COUNTER__ se calcula de la forma siguiente: • el valor inicial del contador es 0, • después de cada uso del contador, su valor aumenta en 1, • primero, el compilador expande en su sitio todas las macros y plantillas en el código fuente, • para cada especialización de la función de plantilla, se genera un código aparte, • para cada especialización de la clase/estructura de plantilla, se genera un código aparte, • a continuación, el compilador recorre el código fuente obtenido en el orden establecido y sustituye cada uso de __COUNTER__ encontrado por el valor actual del contador. En el ejemplo de abajo, podemos ver con claridad cómo el compilador procesa el código fuente y va sustituyendo __COUNTER__ (a medida que encuentra dicho elemento) por valores en aumento secuencial.

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción
__RANDOM__	Para cada declaración __RANDOM__, el compilador coloca en el código un número ulong aleatorio.

Ejemplo:

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- ejemplo de output de información durante la inicialización del Asesor Experto
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//--- definición de intervalos entre los eventos de temporizador
   EventSetTimer(5);
//---
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- ejemplo de output de información durante la deinicialización del Asesor Experto
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//---
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {

//--- output de información durante la llegada de del tick
   Print(" __MQLBUILD__ = ",__MQLBUILD__,"  __FILE__ = ",__FILE__);
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
   test2();
//---
  }
//+------------------------------------------------------------------+
//| test1                                                            |
//+------------------------------------------------------------------+
void test1(string par)
  {
//--- output de información dentro de la función
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__," par=",par);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| test2                                                            |
//+------------------------------------------------------------------+
void test2()
  {
//--- output de información dentro de la función
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
  }
//+------------------------------------------------------------------+
//| OnTimer event handler                                            |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
  }
```

Exemplo para aprender a trabalhar com uma macro __COUNTER__

```
//--- criamos uma macro para impressão rápida da expressão e seu valor no log
#define print(expr) Print(#expr,"=",expr)

//--- definimos uma macro customizada MACRO_COUNTER por meio da macro predefinida __C
#define MACRO_COUNTER __COUNTER__

//--- definimos o valor da variável input com a macro __COUNTER__
input int InpVariable = __COUNTER__;

//--- definimos o valor da variável global com a macro __COUNTER__ antes de definir as
int ExtVariable = __COUNTER__;

//+------------------------------------------------------------------+
//| a função retorna o valor __COUNTER__                             |
//+------------------------------------------------------------------+
int GlobalFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| função de modelo retorna valor __COUNTER__                       |
//+------------------------------------------------------------------+
template<typename T>
int GlobalTemplateFunc(void)
  {
   return(__COUNTER__);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| estrutura com um método que retorna __COUNTER__                  |
//+------------------------------------------------------------------+
struct A
  {
   int               dummy;  // não é usado

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| estrutura baseada em modelo com um método que retorna __COUNTER__|
//+------------------------------------------------------------------+
template<typename T>
struct B
  {
   int               dummy;  // não é usado

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| estrutura com um método baseado em modelo que retorna __COUNTER__|
//+------------------------------------------------------------------+
struct C
  {
   int               dummy;  // não é usado

   template<typename T>
   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| função #2, que retorna o valor __COUNTER__                       |
//+------------------------------------------------------------------+
int GlobalFunc2(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
// __COUNTER__ em macros e variáveis
   print(MACRO_COUNTER);
   print(InpVariable);
   print(ExtVariable);

/--- __COUNTER__ em funções
   print(GlobalFunc());
   print(GlobalFunc());                // o valor não muda
   print(GlobalTemplateFunc<int>());
   print(GlobalTemplateFunc<int>());   // o valor não muda
   print(GlobalTemplateFunc<double>());// o valor mudou
   print(GlobalFunc2());
   print(GlobalFunc2());               // o valor não muda

// __COUNTER__ na estrutura
   A a1, a2;
   print(a1.Method());
   print(a2.Method());                 // o valor não muda

// __COUNTER__ na estrutura baseada em modelo
   B<int> b1, b2;
   B<double> b3;
   print(b1.Method());
   print(b2.Method());                 // o valor não muda
   print(b3.Method());                 // o valor mudou

// __COUNTER__ em uma estrutura com uma função baseada em modelo
   C c1, c2;
   print(c1.Method<int>());
   print(c1.Method<double>());         // o valor mudou
   print(c2.Method<int>());            // valor como ao chamar c1.Method<int>() pela p

//--- mais uma vez, olhamos para __COUNTER__ na macro e na variável global
   print(MACRO_COUNTER);  // o valor mudou
   print(ExtGlobal2);
  }
//--- definimos o valor da variável global com a macro __COUNTER__ após as definições
int ExtGlobal2 = __COUNTER__;
//+------------------------------------------------------------------+

/* Resultado
   __COUNTER__=3
   InpVariable=0
   ExtVariable=1
   GlobalFunc()=5
   GlobalFunc()=5
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<int>()=8
```

© 2000-2025, MetaQuotes Ltd.

```
   GlobalTemplateFunc<double>()=9
   GlobalFunc2()=7
   GlobalFunc2()=7
   a1.Method()=6
   a2.Method()=6
   b1.Method()=10
   b2.Method()=10
   b3.Method()=11
   c1.Method<int>()=12
   c1.Method<double>()=13
   c2.Method<int>()=12
   __COUNTER__=4
   ExtGlobal2=2

*/
```

© 2000-2025, MetaQuotes Ltd.

Constantes matemáticas

Las constantes especiales que contienen valores están reservadas para algunas expresiones matemáticas. Se puede usar estas constantes en cualquier parte del programa mql5 en vez de calcular sus valores a través de las funciones matemáticas.

Constante	Descripción	Valor
M_E	e	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.43429448190325182765 1
M_LN2	ln(2)	0.69314718055994530941 7
M_LN10	ln(10)	2.30258509299404568402
M_PI	pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.78539816339744830962
M_1_PI	1/pi	0.31830988618379067153 8
M_2_PI	2/pi	0.63661977236758134307 6
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.70710678118654752440 1

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- devolvemos los valores de las constantes
   Print("M_E = ",DoubleToString(M_E,16));
   Print("M_LOG2E = ",DoubleToString(M_LOG2E,16));
   Print("M_LOG10E = ",DoubleToString(M_LOG10E,16));
   Print("M_LN2 = ",DoubleToString(M_LN2,16));
   Print("M_LN10 = ",DoubleToString(M_LN10,16));
   Print("M_PI = ",DoubleToString(M_PI,16));
   Print("M_PI_2 = ",DoubleToString(M_PI_2,16));
   Print("M_PI_4 = ",DoubleToString(M_PI_4,16));
   Print("M_1_PI = ",DoubleToString(M_1_PI,16));
   Print("M_2_PI = ",DoubleToString(M_2_PI,16));
   Print("M_2_SQRTPI = ",DoubleToString(M_2_SQRTPI,16));
   Print("M_SQRT2 = ",DoubleToString(M_SQRT2,16));
   Print("M_SQRT1_2 = ",DoubleToString(M_SQRT1_2,16));
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Constantes de tipos numéricos de datos

Cada tipo numérico simple sirve para un cierto grupo de tareas y permite optimizar el funcionamiento de un programa mql5, siempre y cuando se use correctamente. Para la mejor legibilidad del código y procesamiento correcto de los resultados de calculación, existen unas constantes que permiten recibir la información sobre las restricciones establecidas para uno u otro tipo de datos simples.

Constante	Descripción	Valor
CHAR_MIN	Valor mínimo que puede ser representado por el tipo char	-128
CHAR_MAX	Valor máximo que puede ser representado por el tipo char	127
UCHAR_MAX	Valor máximo que puede ser representado por el tipo uchar	255
SHORT_MIN	Valor mínimo que puede ser representado por el tipo short	-32768
SHORT_MAX	Valor máximo que puede ser representado por el tipo short	32767
USHORT_MAX	Valor máximo que puede ser representado por el tipo ushort	65535
INT_MIN	Valor mínimo que puede ser representado por el tipo int	-2147483648
INT_MAX	Valor máximo que puede ser representado por el tipo int	2147483647
UINT_MAX	Valor máximo que puede ser representado por el tipo uint	4294967295
LONG_MIN	Valor mínimo que puede ser representado por el tipo long	-9223372036854775808
LONG_MAX	Valor máximo que puede ser representado por el tipo long	9223372036854775807
ULONG_MAX	Valor máximo que puede ser representado por el tipo ulong	18446744073709551615
DBL_MIN	Valor mínimo positivo que puede ser representado por el tipo double	2.2250738585072014e-308
DBL_MAX	Valor máximo que puede ser representado por el tipo double	1.7976931348623158e+308
DBL_EPSILON	Valor mínimo que satisface la condición: 1.0+DBL_EPSILON != 1.0	2.2204460492503131e-016
DBL_DIG	Número de dígitos decimales significativos	15

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Valor
DBL_MANT_DIG	Cantidad de bits en la mantisa	53
DBL_MAX_10_EXP	Valor decimal máximo del grado de exponente	308
DBL_MAX_EXP	Valor binario máximo del grado de exponente	1024
DBL_MIN_10_EXP	Valor decimal máximo del grado de exponente	(-307)
DBL_MIN_EXP	Valor binario mínimo del grado de exponente	(-1021)
FLT_MIN	Valor mínimo positivo que puede ser representado por el tipo float	1.175494351e-38
FLT_MAX	Valor máximo que puede ser representado por el tipo float	3.402823466e+38
FLT_EPSILON	Valor mínimo que satisface la condición: 1.0+FLT_EPSILON != 1.0	1.192092896e-07
FLT_DIG	Número de dígitos decimales significativos	6
FLT_MANT_DIG	Cantidad de bits en la mantisa	24
FLT_MAX_10_EXP	Valor decimal máximo del grado de exponente	38
FLT_MAX_EXP	Valor binario máximo del grado de exponente	128
FLT_MIN_10_EXP	Valor decimal mínimo del grado de exponente	-37
FLT_MIN_EXP	Valor binario mínimo del grado de exponente	(-125)

Ejemplo:

```
void OnStart()
  {
//--- devolvemos los valores de las constantes
   printf("CHAR_MIN = %d",CHAR_MIN);
   printf("CHAR_MAX = %d",CHAR_MAX);
   printf("UCHAR_MAX = %d",UCHAR_MAX);
   printf("SHORT_MIN = %d",SHORT_MIN);
   printf("SHORT_MAX = %d",SHORT_MAX);
   printf("USHORT_MAX = %d",USHORT_MAX);
   printf("INT_MIN = %d",INT_MIN);
   printf("INT_MAX = %d",INT_MAX);
   printf("UINT_MAX = %u",UINT_MAX);
```

© 2000-2025, MetaQuotes Ltd.

```
   printf("LONG_MIN = %I64d",LONG_MIN);
   printf("LONG_MAX = %I64d",LONG_MAX);
   printf("ULONG_MAX = %I64u",ULONG_MAX);
   printf("EMPTY_VALUE = %.16e",EMPTY_VALUE);
   printf("DBL_MIN = %.16e",DBL_MIN);
   printf("DBL_MAX = %.16e",DBL_MAX);
   printf("DBL_EPSILON = %.16e",DBL_EPSILON);
   printf("DBL_DIG = %d",DBL_DIG);
   printf("DBL_MANT_DIG = %d",DBL_MANT_DIG);
   printf("DBL_MAX_10_EXP =  %d",DBL_MAX_10_EXP);
   printf("DBL_MAX_EXP = %d",DBL_MAX_EXP);
   printf("DBL_MIN_10_EXP = %d",DBL_MIN_10_EXP);
   printf("DBL_MIN_EXP = %d",DBL_MIN_EXP);
   printf("FLT_MIN = %.8e",FLT_MIN);
   printf("FLT_MAX = %.8e",FLT_MAX);
   printf("FLT_EPSILON = %.8e",FLT_EPSILON);
/*
   CHAR_MIN = -128
   CHAR_MAX = 127
   UCHAR_MAX = 255
   SHORT_MIN = -32768
   SHORT_MAX = 32767
   USHORT_MAX = 65535
   INT_MIN = -2147483648
   INT_MAX = 2147483647
   UINT_MAX = 4294967295
   LONG_MIN = -9223372036854775808
   LONG_MAX = 9223372036854775807
   ULONG_MAX = 18446744073709551615
   EMPTY_VALUE = 1.7976931348623157e+308
   DBL_MIN = 2.2250738585072014e-308
   DBL_MAX = 1.7976931348623157e+308
   DBL_EPSILON = 2.2204460492503131e-16
   DBL_DIG = 15
   DBL_MANT_DIG = 53
   DBL_MAX_10_EXP =  308
   DBL_MAX_EXP = 1024
   DBL_MIN_10_EXP = -307
   DBL_MIN_EXP = -1021
   FLT_MIN = 1.17549435e-38
   FLT_MAX = 3.40282347e+38
   FLT_EPSILON = 1.19209290e-07
*/
  }
```

© 2000-2025, MetaQuotes Ltd.

Razones de deinicialización

Los códigos de las razones de deinicialización del Asesor Experto devueltos por la función UninitializeReason(). Pueden tener cualquier de los siguientes valores:

Constante	Valor	Descripción
REASON_PROGRAM	0	Asesor Experto finalizó su trabajo llamando a la función ExpertRemove()
REASON_REMOVE	1	El programa ha sido eliminado del gráfico
REASON_RECOMPILE	2	El programa ha sido recompilado
REASON_CHARTCHANGE	3	El símbolo o período del gráfico ha sido modificado
REASON_CHARTCLOSE	4	El gráfico ha sido cerrado
REASON_PARAMETERS	5	Los parámetros de entrada han sido cambiados por el usuario
REASON_ACCOUNT	6	Ha sido activada otra cuenta o ha tenido lugar la reconexión con el servidor comercial debido al cambio de los ajustes de la cuenta
REASON_TEMPLATE	7	Ha sido aplicada la nueva plantilla del gráfico
REASON_INITFAILED	8	Este valor significa que el manejador OnInit() ha devuelto un valor no nulo
REASON_CLOSE	9	El terminal ha sido cerrado

El código de la razón de deinicialización se pasa también como un parámetro de la función predefinida OnDeinit(const int reason).

Ejemplo:

```
//+------------------------------------------------------------------+
//| get text description                                             |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Account was changed";break;
      case REASON_CHARTCHANGE:
         text="Symbol or timeframe was changed";break;
      case REASON_CHARTCLOSE:
         text="Chart was closed";break;
      case REASON_PARAMETERS:
         text="Input-parameter was changed";break;
      case REASON_RECOMPILE:
```

© 2000-2025, MetaQuotes Ltd.

```
         text="Program "+__FILE__+" was recompiled";break;
      case REASON_REMOVE:
         text="Program "+__FILE__+" was removed from chart";break;
      case REASON_TEMPLATE:
         text="New template was applied to chart";break;
      default:text="Another reason";
     }
//---
   return text;
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- El primer modo de obtener el código de la razón de deinicialización
   Print(__FUNCTION__,"_Código de la razón de deinicialización = ",reason);
//--- El segundo modo de obtener el código de la razón de deinicialización
   Print(__FUNCTION__,"_UninitReason = ",getUninitReasonText(_UninitReason));
  }
```

© 2000-2025, MetaQuotes Ltd.

Verificación del puntero a objeto

La función CheckPointer() sirve para comprobar el tipo del puntero a objeto. Esta función devuelve el valor de la enumeración ENUM_POINTER_TYPE. En caso de usar un puntero incorrecto, la ejecución del programa se detendrá inmediatamente.

Los objetos creados por el operador new son del tipo POINTER_DYNAMIC. Sólo para estos punteros se puede y se debe usar el operador de eliminación delete().

Todos los demás punteros tienen el tipo POINTER_AUTOMATIC, lo que significa que este objeto ha sido creado automáticamente por el entorno del programa mql5. Estos objetos se eliminan después de ser usados también de una manera automática.

ENUM_POINTER_TYPE

Constante	Descripción
POINTER_INVALID	Puntero incorrecto
POINTER_DYNAMIC	Puntero a objeto que ha sido creado por el operador new
POINTER_AUTOMATIC	Puntero a cualquier objeto que ha sido creado automáticamente (sin usar new())

Véase también

Errores de ejecución, Operador de eliminación de objeto delete, CheckPointer()

© 2000-2025, MetaQuotes Ltd.

Otras constantes

La constante CLR_NONE sirve para indicar la falta del color, es decir, el objeto gráfico o serie gráfica de un indicador no serán mostrados. Esta constante no ha entrado en la lista de constantes de colores Web pero se puede usarla en cualquier parte donde se requiere indicar un color.

La constante INVALID_HANDLE puede ser usada durante la depuración de los manejadores de archivos(véase FileOpen() y FileFindFirst()).

Constante	Descripción	Valor
CHARTS_MAX	La cantidad máxima posible de los gráficos abiertos al mismo tiempo en el terminal	100
clrNONE	Ausencia de color	-1
EMPTY_VALUE	Valor vacío en el búfer de indicadores	DBL_MAX
INVALID_HANDLE	Manejador incorrecto	-1
IS_DEBUG_MODE	Indica que un programa mql5 se encuentra en el modo de depuración	true en el modo de depuración, de lo contrario false
IS_PROFILE_MODE	Indica que un programa mql5 se encuentra en el modo de perfilación	en modo de perfilación no es igual a cero, de lo contrario 0
NULL	Cero de cualquier tipo	0
WHOLE_ARRAY	Significa el número de elementos que se quedan hasta el final del array, es decir, el array entero será procesado	-1
WRONG_VALUE	La constante puede convertirse implícitamente al tipo de cualquier enumeración.	-1

La constante EMPTY_VALUE suele corresponder a los valores de los indicadores que no se muestran en el gráfico. Por ejemplo, para el indicador built-in Standard Deviation con el período 20, la línea para las primeras 19 barras en el historial no se muestra en el gráfico. Si creamos el manejador de este indicador usando la función iStdDev() y copiamos en el array los valores del indicador para estas baras a través de CopyBuffer(), entonces precisamente estos valores serán iguales a EMPTY_VALUE.

Nosotros mismos podemos especificar nuestro propio valor vacío del indicador en el indicador personalizado, en este caso el indicador no debería mostrarse en el gráfico. Con este fin se usa la función PlotIndexSetDouble() con el modificador PLOT_EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

La constante NULL puede ser asignada a una variable de cualquier tipo simple o a un puntero a objeto de estructura o clase. La asignación de NULL a una variable de cadena significa la deinicialización completa de esta variable.

La constante WRONG_VALUE sirve para los casos cuando hace falta devolver el valor de una enumeración, y éste tiene que ser un valor erróneo. Por ejemplo, cuando tenemos que informar que un valor devuelto es un valor de esta enumeración. Como ilustración vamos a considerar una función CheckLineStyle() que devuelve el estilo de la línea para un objeto especificado por su nombre. Si el resultado de comprobación del estilo por la función ObjectGetInteger() es true, entonces será devuelto el valor de la enumeración ENUM_LINE_STYLE, de lo contrario se devuelve WRONG_VALUE.

```
void OnStart()
  {
   if(CheckLineStyle("MyChartObject")==WRONG_VALUE)
      printf("Error line style getting.");
  }
//+------------------------------------------------------------------+
//| devuelve el estilo de la línea de un objeto especificado por su nombre          |
//+------------------------------------------------------------------+
ENUM_LINE_STYLE CheckLineStyle(string name)
  {
   long style;
//---
   if(ObjectGetInteger(0,name,OBJPROP_STYLE,0,style))
      return((ENUM_LINE_STYLE)style);
   else
      return(WRONG_VALUE);
  }
```

La constante WHOLE_ARRAY está destinada para las funciones que requieren la especificación de la cantidad de elementos en los arrays procesados:

- ArrayCopy();
- ArrayMinimum();
- ArrayMaximum();
- FileReadArray();
- FileWriteArray().

Si hace falta especificar que es necesario procesar todos los valores del array desde la posición especificada hasta el final, será suficiente indicar el valor WHOLE_ARRAY.

La constante IS_PROFILE_MODE permite cambiar el trabajo del programa para la correcta recopilación de la información en el modo de perfilación. La perfilación permite medir el tiempo de ejecución de ciertos fragmentos del programa (normalmente son las funciones), así como contar el número de estas llamadas. Para una correcta obtención de información sobre el tiempo de ejecución en el modo de perfilación se puede desactivar las llamadas a la función Sleep() como se muestra en el ejemplo:

```
//--- Sleep puede influir (desfigurar) considerablemente en el resultado de la perfila
if(!IS_PROFILE_MODE) Sleep(100); // prohibimos la llamada a Sleep() en el modo de peri
```

El valor de la constante IS_PROFILE_MODE se establece por el compilador en el momento de compilación, mientras que en el modo convencional se pone igual a cero. Cuando el programa se inicia

© 2000-2025, MetaQuotes Ltd.

en el modo de perfilación, se lleva a cabo una compilación especial, y en este caso IS_PROFILE_MODE se sustituye con un valor distinto a cero.

La constante IS_DEBUG_MODE será útil cuando es necesario cambiar un poco el trabajo de un programa mql5 en el modo de depuración. Por ejemplo, durante la depuración surge la necesidad de mostrar la información adicional de depuración en el registro (log) del terminal o crear los objetos gráficos auxiliares en un gráfico.

El ejemplo de abajo crea un objeto Label y define su descripción y color dependiendo del régimen en el que se ejecute el script. Para iniciar un script en el modo de depuración en MetaEditor, presione el botón F5. Si iniciamos el script desde la ventana del navegador en el terminal, el color y el texto del objeto Label serán diferentes.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                              Check_DEBUG_MODE.mq5 |
//|                        Copyright © 2009, MetaQuotes Software Corp. |
//|                                             https://www.metaquotes.net |
//+------------------------------------------------------------------+
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="invisible_label";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Object ",label_name," not found. Error code = ",GetLastError());
      //--- creamos el objeto Label
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- establecemos la coordinada X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- establecemos la coordinada Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      ResetLastError();
      if(IS_DEBUG_MODE) // modo de depuración
        {
         //--- mostramos el mensaje sobre el modo de ejecución del script
         ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
         //--- establecemos el color del texto como rojo
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
            Print("Fallo al determinar el color. Error ",GetLastError());
        }
      else              // modo operacional
        {
         ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
         //--- fijamos el color invisible del texto
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
            Print("Fallo al determinar el color. Error ",GetLastError());
        }
      ChartRedraw();
      DebugBreak();    // si nos encontramos en el modo de depuración, aquí sucederá
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Métodos del cifrado de datos

Para indicar el método de conversión de datos (cifrado y cálculo de hashs), en las funciones CryptEncode() y CryptDecode() se usa la enumeración ENUM_CRYPT_METHOD.

ENUM_CRYPT_METHOD

Constante	Descripción
CRYPT_BASE64	Cifrado BASE64 (recodificación)
CRYPT_AES128	Cifrado AES con la clave de 128 bits (16 bytes)
CRYPT_AES256	Cifrado AES con la clave de 256 bits (32 bytes)
CRYPT_DES	Cifrado DES con la clave de 56 bits (7 bytes)
CRYPT_HASH_SHA1	Cálculo HASH SHA1
CRYPT_HASH_SHA256	Cálculo HASH SHA256
CRYPT_HASH_MD5	Cálculo HASH MD5
CRYPT_ARCH_ZIP	Compresión ZIP

Véase también

DebugBreak, Información sobre el programa MQL5 en ejecución, CryptEncode(), CryptDecode()

© 2000-2025, MetaQuotes Ltd.

Estructuras de datos

En MQL5 hay 12 estructuras predefinidas que sirven para el almacenamiento y traspaso de información auxiliar:

- MqlDateTime sirve para representar fecha y hora;
- MqlParam permite pasar los parámetros de entrada durante la creación de un manejador (handle) de indicador utilizando la función IndicatorCreate();
- MqlRates se usa para facilitar la información sobre los datos históricos que contienen el precio, volumen y spread;
- MqlBookInfo se usa para obtener información reflejada en la profundidad de mercado (ventana de cotizaciones);
- MqlTradeRequest se usa para crear una orden comercial durante las operaciones comerciales;
- MqlTradeResult contiene la respuesta del servidor comercial a una orden comercial mandada por la función OrderSend();
- MqlTradeCheckResult permite comprobar la orden comercial preparada antes de enviarla;
- MqlTradeTransaction contiene la descripción de transacción comercial;
- MqlTick sirve para la obtención rápida de la información más requerida sobre los precios actuales.
- Las estructuras del Calendario Económico han sido diseñadas para obtener información sobre los eventos del Calendario Económico que llegan a la plataforma MetaTrader en tiempo real. Las funciones del Calendario Económico permiten analizar los indicadores macroeconómicos justo tras la publicación de los nuevos informes, ya que los valores actuales se retransmiten directamente desde la fuente, sin retardos.

© 2000-2025, MetaQuotes Ltd.

MqlDateTime

La estructura de la fecha contiene ocho campos del tipo int.

```
struct MqlDateTime
  {
   int year;            // año
   int mon;             // mes
   int day;             // día
   int hour;            // hora
   int min;             // minutos
   int sec;             // segundos
   int day_of_week;     // día de la semana (0-domingo, 1-lunes, ... ,6-sábado)
   int day_of_year;     // número del día del año (el primero de enero tiene el número 0)
  };
```

Nota

Número del día del año day_of_year del año bisiesto, empezando desde Marzo, será diferente del número del día del año en el año no bisiesto.

Ejemplo:

```
void OnStart()
  {
//---
   datetime date1=D'2008.03.01';
   datetime date2=D'2009.03.01';

   MqlDateTime str1,str2;
   TimeToStruct(date1,str1);
   TimeToStruct(date2,str2);
   printf("%02d.%02d.%4d, day of year = %d",str1.day,str1.mon,
          str1.year,str1.day_of_year);
   printf("%02d.%02d.%4d, day of year = %d",str2.day,str2.mon,
          str2.year,str2.day_of_year);
  }
/*  Resultado
   01.03.2008, day of year = 60
   01.03.2009, day of year = 59
*/
```

Véase también

TimeToStruct, Estructuras y clases

© 2000-2025, MetaQuotes Ltd.

Estructura de parámetros de entrada de indicador (MqlParam)

La estructura MqlParam ha sido diseñada especialmente para traspasar los parámetros de entrada cuando se crea el manejador del indicador técnico usando la función IndicatorCreate().

```
struct MqlParam
  {
   ENUM_DATATYPE     type;                    // tipo del parámetro de entrada, valor de la enumerac
   long              integer_value;           // campo para almacenar valores de números enteros
   double            double_value;            // campo para almacenar valores double o float
   string            string_value;            // campo para almacenar valores del tipo string
  };
```

Todos los parámetros de entrada se transmiten en forma de una matriz del tipo MqlParam, el campo *type* de cada uno de los elementos de esta matriz especifica el tipo de datos que transmite dicho elemento. Previamente hay que colocar los valores de parámetros del indicador en los campos correspondientes para cada elemento (en *integer_value*, en *double_value* o en *string_value*), dependiendo de qué valor de la enumeración ENUM_DATATYPE figura en el campo *type*.

Si el valor IND_CUSTOM en el tercer parámetro se pasa a la función IndicatorCreate() como el tipo de indicador, entonces el primer elemento de la matriz de los parámetros de entrada debe tener el campo *type* con el valor TYPE_STRING de la enumeración ENUM_DATATYPE, y el campo *string_value* tiene que contener el nombre del indicador personalizado.

© 2000-2025, MetaQuotes Ltd.

MqlRates

Esta estructura sirve para almacenar la información sobre los precios, volúmenes y spread.

```
struct MqlRates
  {
   datetime time;         // hora del inicio del período
   double   open;         // precio de apertura
   double   high;         // precio máximo durante el período
   double   low;          // precio mínimo durante el período
   double   close;        // precio de cierre
   long     tick_volume;  // volumen de tick
   int      spread;       // spread
   long     real_volume;  // volumen de stock
  };
```

Ejemplo:

```
void OnStart()
  {
   MqlRates rates[];
   int copied=CopyRates(NULL,0,0,100,rates);
   if(copied<=0)
      Print("Fallo al copiar los datos de precios ",GetLastError());
   else Print("Se ha copiado ",ArraySize(rates)," barras");
  }
```

Véase también

CopyRates, Acceso a series temporales

© 2000-2025, MetaQuotes Ltd.

MqlBookInfo

Esta estructura proporciona la información sobre la profundidad de mercado.

```
struct MqlBookInfo
  {
   ENUM_BOOK_TYPE    type;          // tipo de orden desde la enumeración ENUM_BOOK_T
   double            price;         // precio
   long              volume;        // volumen
   double            volume_real;   // volumen con precisión aumentada
  };
```

Nota

La estructura MqlBookInfo es predefinida, por eso no hace falta declarar y describirla. Para utilizar la estructura, será suficiente declarar la variable de este tipo.

La profundidad de mercado está disponible sólo para algunos instrumentos financieros.

Ejemplo:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo para ",Symbol());
     }
   else
     {
      Print("Fallo al recibir el contenido de la profundidad de mercado para el símbol
     }
```

Véase también

MarketBookAdd, MarketBookRelease, MarketBookGet, Tipos de órdenes en profundidad de mercado, Tipos de datos

© 2000-2025, MetaQuotes Ltd.

Estructura de solicitud comercial (MqlTradeRequest)

La interacción entre el terminal de cliente y el servidor comercial con el fin de ejecutar las operaciones de colocación de las órdenes se realiza a través de las solicitudes comerciales. La solicitud comercial está representada por la estructura especial predefinida MqlTradeRequest que contiene todos los campos necesarios para celebrar las transacciones comerciales. El resultado de procesamiento de una solicitud está representado por la estructura MqlTradeResult.

```
struct MqlTradeRequest
  {
   ENUM_TRADE_REQUEST_ACTIONS    action;           // Tipo de acción que se ejecuta
   ulong                         magic;            // ID del Asesor Experto (identificador magic num
   ulong                         order;            // Ticket de la orden
   string                        symbol;           // Nombre del instrumento comercial
   double                        volume;           // Volumen solicitado de la transacción en lotes
   double                        price;            // Precio
   double                        stoplimit;        // Nivel StopLimit de la orden
   double                        sl;               // Nivel Stop Loss de la orden
   double                        tp;               // Nivel Take Profit de la orden
   ulong                         deviation;        // Desviación máxima aceptable del precio solicit
   ENUM_ORDER_TYPE               type;             // Tipo de orden
   ENUM_ORDER_TYPE_FILLING       type_filling;     // Tipo de ejecución de la orden
   ENUM_ORDER_TYPE_TIME          type_time;        // Tipo de orden por su plazo de ejecución
   datetime                      expiration;       // Plazo de expiración de la orden (para las órde
   string                        comment;          // Comentarios sobre la orden
   ulong                         position;         // Position ticket
   ulong                         position_by;      // Comentarios sobre la orden
  };
```

Descripción de campos

Campo	Descripción
action	Tipo de operación comercial. El valor puede ser uno de los valores de la enumeración ENUM_TRADE_REQUEST_ACTIONS
magic	Identificador del Asesor Experto. Permite organizar el procesamiento analítico de las órdenes comerciales. Cada Asesor Experto puede establecer su propio identificador personal único a la hora de mandar una solicitud comercial.
order	Ticket de la orden. Se necesita para modificar las órdenes pendientes.
symbol	Nombre del instrumento financiero (símbolo) para el que se coloca una orden. No se necesita para modificar las órdenes y cerrar las posiciones.
volume	Volumen solicitado de la transacción en lotes. El valor real del volumen dependerá del tipo de ejecución de la orden.
price	Precio. Al alcanzarlo, la orden tiene que ejecutarse. Las órdenes del mercado de símbolos, cuyo tipo de ejecución es "Market Execution" (SYMBOL_TRADE_EXECUTION_MARKET), del tipo TRADE_ACTION_DEAL, no requieren la especificación del precio.
stoplimit	Precio según el cual será colocado la orden pendiente Limit, cuando el precio alcance el valor price (esta condición es obligatoria). Hasta entonces la orden pendiente no se introduce en el sistema.

© 2000-2025, MetaQuotes Ltd.

Campo	Descripción
sl	Precio que activará la orden Stop Loss, si el precio se mueve en la dirección desfavorable
tp	Precio que activará la orden Take Profit. si el precio se mueve en la dirección favorable
deviation	Desviación máxima aceptable del precio solicitado, se especifica en puntos
type	Tipo de la orden. Su valor puede ser uno de los valores de la enumeración ENUM_ORDER_TYPE
type_filling	Tipo de ejecución de la orden. Su valor puede ser uno de los valores de ENUM_ORDER_TYPE_FILLING
type_time	Tipo de orden por su plazo de ejecución. Su valor puede ser uno de los valores de ENUM_ORDER_TYPE_TIME
expiration	Plazo de expiración de la orden (para las órdenes del tipo ORDER_TIME_SPECIFIED)
comment	Comentarios sobre la orden

Al modificar o cerrar una posición en el sistema de cobertura, asegúrese de indicar su ticket (MqlTradeRequest::position). En el sistema de compensación el ticket también se puede indicar, sin embargo, la identificación de la posición tiene lugar según el nombre del símbolo.

Para dar las órdenes de ejecución de las operaciones comerciales es necesario usar la función OrderSend(). Para cada operación comercial hay que indicar los campos obligatorios, también se puede rellenar campos opcionales. En total hay siete formas de enviar una solicitud comercial:

Request Execution

Es una orden comercial para abrir una posición en el régimen Request Execution (régimen de actividad comercial sobre solicitud de precios actuales). Se requiere especificar 9 campos:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

Además se puede definir los valores de los campos magic y comment.

Instant Execution

Es una orden comercial para abrir una posición en el régimen Instant Execution (régimen de actividad comercial a base de los precios corrientes). Se requiere especificar 9 campos:

- action

© 2000-2025, MetaQuotes Ltd.

- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

Además se puede definir los valores de los campos magic comment.

Market Execution

Es una orden comercial para abrir una posición en el régimen Market Execution (régimen de ejecución de las órdenes comerciales en el mercado). Se requiere especificar 5 campos:

- action
- symbol
- volume
- type
- type_filling

Además se puede definir los valores de los campos magic y comment.

Exchange Execution

Es una orden comercial para abrir una posición en el régimen Exchange Execution (modo de ejecución de órdenes comerciales por bolsa). Se requiere especificar 5 campos:

- action
- symbol
- volume
- type
- type_filling

Además se puede definir los valores de los campos magic y comment.

Ejemplo de la operación comercial TRADE_ACTION_DEAL para abrir una posición Buy:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Apertura de posición Buy                                         |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración e inicialización de la solicitud y el resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parámetros de la solicitud
   request.action   =TRADE_ACTION_DEAL;                  // tipo de operación comer
   request.symbol   =Symbol();                           // símbolo
   request.volume   =0.1;                                // volumen de 0.1 lote
   request.type     =ORDER_TYPE_BUY;                     // tipo de orden
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // precio de apertura
   request.deviation=5;                                  // desviación permisible d
   request.magic    =EXPERT_MAGIC;                       // Número mágico de la ord
//--- envío de la solicitud
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // si no se ha logrado env
//--- información sobre la operación
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Ejemplo de la operación comercial TRADE_ACTION_DEAL para abrir una posición Sell:

```
#define EXPERT_MAGIC 123456   // Número mágico del experto
//+------------------------------------------------------------------+
//| Apertura de posición Sell                                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración e inicialización de la solicitud y el resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parámetros de la solicitud
   request.action   =TRADE_ACTION_DEAL;                  // tipo de operación comer
   request.symbol   =Symbol();                           // símbolo
   request.volume   =0.2;                                // volumen 0.2 lote
   request.type     =ORDER_TYPE_SELL;                    // tipo de orden
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // precio de apertura
   request.deviation=5;                                  // desviación permisible d
   request.magic    =EXPERT_MAGIC;                       // Número mágico de la ord
//--- envío de la solicitud
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // si no se ha logrado env
//--- información sobre la operación
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Ejemplo de la operación comercial TRADE_ACTION_DEAL para cerrar una posición:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // Número mágico del experto
//+------------------------------------------------------------------+
//| Cierre de todas las posiciones                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración de la solicitud y el resultado
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // número de posiciones abiertas
//--- iteración de todas las posiciones abiertas
   for(int i=total-1; i>=0; i--)
     {
      //--- parámetros de la orden
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- mostrar la información de la posición
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- si el número mágico coincide
      if(magic==EXPERT_MAGIC)
        {
         //--- reseteo de los valores de la solicitud y el resultado
         ZeroMemory(request);
         ZeroMemory(result);
         //--- establecer los parámetros de la operación
         request.action   =TRADE_ACTION_DEAL;        // tipo de operación comercial
         request.position =position_ticket;          // ticket de la posición
         request.symbol   =position_symbol;          // símbolo
         request.volume   =volume;                   // volumen de la posición
         request.deviation=5;                        // desviación permisible del prec
         request.magic    =EXPERT_MAGIC;             // Número mágico de la posición
         //--- establecer el precio y el tipo de orden dependiendo del tipo de posici
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
           }
         //--- mostrar información sobre el cierre
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- envío de la solicitud
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // si no se ha logrado
         //--- información sobre la operación
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
//+------------------------------------------------------------------+
```

SL & TP Modification

Es una orden comercial para modificar los niveles StopLoss y/o TakeProfit. Se requiere especificar 4 campos:

- action
- symbol
- sl
- tp

Ejemplo de la operación comercial TRADE_ACTION_SLTP para cambiar los valores de Stop Loss y Take Profit de una posición abierta:

```
#define EXPERT_MAGIC 123456  // Número mágico del experto
//+------------------------------------------------------------------+
//| Modificación de Stop Loss y Take Profit de la posición           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración de la solicitud y el resultado
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // número de posiciones abiertas
//--- iteración de todas las posiciones abiertas
   for(int i=0; i<total; i++)
     {
      //--- parámetros de la orden
      ulong  position_ticket=PositionGetTicket(i);// ticket de la posición
      string position_symbol=PositionGetString(POSITION_SYMBOL); // símbolo
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // número d
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // Número mágico de la posición
      double volume=PositionGetDouble(POSITION_VOLUME);    // volumen de la posición
      double sl=PositionGetDouble(POSITION_SL);  // Stop Loss de la posición
      double tp=PositionGetDouble(POSITION_TP);  // Take Profit de la posición
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- mostrar información de la posición
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- si el número mágico coincide, Stop Loss y Take Profit no han sido estable
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- cálculo de los niveles de precio actuales
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_L
         double price_level;
         //--- si el nivel de separación máxima permitida en puntos con respecto al pr
         if(stop_level<=0)
            stop_level=150; // establecemos una separación de 150 puntos con respecto
         else
            stop_level+=50; // tomaremos un nivel de separación igual a (SYMBOL_TRADE_

         //--- cálculo y redondeo de los valores de Stop Loss y Take Profit
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(bid+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(ask-price_level,digits);
           }
         //--- reseteo de los valores de la solicitud y el resultado
         ZeroMemory(request);
         ZeroMemory(result);
         //--- establecer los parámetros de la operación
         request.action  =TRADE_ACTION_SLTP; // tipo de operación comercial
         request.position=position_ticket;   // ticket de la posición
         request.symbol=position_symbol;     // símbolo
         request.sl      =sl;                // Stop Loss de la posición
         request.tp      =tp;                // Take Profit de la posición
         request.magic=EXPERT_MAGIC;         // Número mágico de la posición
         //--- mostrar información de la modificación
         PrintFormat("Modify #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- envío de la solicitud
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // si no se ha logrado
         //--- información sobre la operación
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Pending Order

Es una orden comercial para colocar una orden pendiente. Se requiere especificar 11 campos:

- action
- symbol
- volume
- price
- stoplimit
- sl
- tp
- type

© 2000-2025, MetaQuotes Ltd.

- type_filling
- type_time
- expiration

Además se puede definir los valores de los campos magic y comment.

Ejemplo de la operación comercial TRADE_ACTION_PENDING para colocar una orden pendiente:

© 2000-2025, MetaQuotes Ltd.

```
#property description "Ejemplo de colocación de órdenes pendientes"
#property script_show_inputs
#define EXPERT_MAGIC 123456                              // Número mágico del experto
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;    // tipo de orden
//+------------------------------------------------------------------+
//| Colocar órdenes pendientes                                       |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración e inicialización de la solicitud y el resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parámetros para la colocación de una orden pendiente
   request.action   =TRADE_ACTION_PENDING;                              // tipo de oper
   request.symbol   =Symbol();                                          // símbolo
   request.volume   =0.1;                                               // volumen 0.1
   request.deviation=2;                                                 // desviación p
   request.magic    =EXPERT_MAGIC;                                      // Número mágic
   int offset = 50;                                                     // distancia co
   double price;                                                        // precio de ac
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);                 // tamaño del p
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);                 // número de dí
   //--- comprobación del tipo de operación
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                          // tipo de orde
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;        // precio de ap
      request.price    =NormalizeDouble(price,digits);                 // precio de ap
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                          // tipo de orc
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;         // precio de a
      request.price    =NormalizeDouble(price,digits);                 // precio de a
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                               // tipo de orc
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point; // precio de a
      request.price=NormalizeDouble(price,digits);                     // precio de a
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                          // tipo de orc
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;        // precio de a
      request.price    =NormalizeDouble(price,digits);                 // precio de a
     }
   else Alert("Este ejemplo es solo para colocar órdenes pendientes");   // si no se h
//--- enviar solicitud
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());                // si no se ha
//--- información sobre la operación
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Modify Pending Order

Es una orden comercial para modificar los precios de una orden pendiente. Se requiere especificar 7 campos:

© 2000-2025, MetaQuotes Ltd.

- action
- order
- price
- sl
- tp
- type_time
- expiration

Ejemplo de la operación comercial TRADE_ACTION_MODIFY para modificar los niveles de precio de las órdenes pendientes:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // Número mágico del experto
//+------------------------------------------------------------------+
//| Modificación de órdenes pendientes                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración e incialización de la solicitud y el resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // número de órdenes pendientes colocadas
//--- iteración de todas las órdenes pendientes colocadas
   for(int i=0; i<total; i++)
     {
      //--- parámetros de la orden
      ulong  order_ticket=OrderGetTicket(i);                          // ticket de
      string order_symbol=Symbol();                                   // símbolo
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS); // número de
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                      // Número mág
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);             // volumen ac
      double sl=OrderGetDouble(ORDER_SL);                             // Stop Loss
      double tp=OrderGetDouble(ORDER_TP);                             // Take Profi
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // tipo de or
      int offset = 50;                                                // distancia
      double price;                                                   // precio de
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);       // tamaño del
      //--- mostrar información sobre la orden
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  order_ticket,
                  order_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- si el número mágico coincide, Stop Loss y Take Profit no han sido establec
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                          // tipo de oper
         request.order = OrderGetTicket(i);                           // ticket de la
         request.symbol   =Symbol();                                  // símbolo
         request.deviation=5;                                         // desviación p
        //--- establecer los niveles de precio, Take Profit y Stop Loss de la orden de
         if(type==ORDER_TYPE_BUY_LIMIT)
           {
            price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // precio
           }
         else if(type==ORDER_TYPE_SELL_LIMIT)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // preci
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
```

© 2000-2025, MetaQuotes Ltd.

```
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // precio
           }
         else if(type==ORDER_TYPE_SELL_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // precio
           }
         //--- envío de la solicitud
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // si no se ha logrado
         //--- información sobre la operación
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //--- reseteo de los valores de la solicitud y el resultado
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

Delete Pending Order

Es una orden comercial para eliminar una orden pendiente. Se requiere especificar 2 campos:

- action
- order

Ejemplo de la operación comercial TRADE_ACTION_REMOVE para eliminar órdenes pendientes:

```
#define EXPERT_MAGIC 123456  // Número mágico del experto
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Eliminación de órdenes pendientes                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaración e inicialización de la solicitud y el resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // cantidad de órdenes pendientes establecidas
//--- iteración de todas las órdenes pendientes establecidas
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // ticket de la orden
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // Número mágico de la
      //--- si el número mágico coincide
      if(magic==EXPERT_MAGIC)
        {
         //--- reseteo de los valores de la solicitud y el resultado
         ZeroMemory(request);
         ZeroMemory(result);
         //--- establecer los parámetros de la operación
         request.action=TRADE_ACTION_REMOVE;                   // tipo de operación co
         request.order = order_ticket;                         // ticket de la orden
         //--- envío de la solicitud
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // si no se ha logrado
         //--- información sobre la operación
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Véase también

Estructuras y clases, Funciones comerciales, Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

Estructura de comprobación de solicitud comercial (MqlTradeCheckResult)

Antes de enviar una solicitud para una operación comercial al servidor comercial, se recomienda efectuar comprobaciones de su viabilidad. Dicha comprobación se realiza usando la función OrderCheck() a la que se pasa la solicitud a comprobar y también la variable del tipo de estructura MqlTradeCheckResult. Precisamente en esta variable será reflejado el resultado del chequeo realizado.

```
struct MqlTradeCheckResult
  {
   uint         retcode;             // Código de respuesta
   double       balance;             // Balance después de realizar la transacción
   double       equity;              // Capital privado después de realizar la transacción
   double       profit;              // Beneficio flotante
   double       margin;              // Requerimientos de margen
   double       margin_free;         // Margen libre
   double       margin_level;        // Nivel del margen
   string       comment;             // Comentarios sobre el código de respuesta (descripción del er
  };
```

Descripción de campos

Campo	Descripción
retcode	Código de retorno
balance	Balance que se queda tras la ejecución de la operación comercial
equity	Valor de fondos propios que se obtiene tras la ejecución de la operación comercial
profit	Valor del beneficio flotante que se obtiene tras la ejecución de la operación comercial
margin	Margen necesario para la operación comercial
margin_free	Fondos propios que se quedan disponibles después de realizar la operación comercial
margin_level	Nivel del margen que va a establecerse después de realizar la operación comercial
comment	Comentario sobre el código de respuesta, descripción del error en su caso

Véase también

Estructura de solicitud comercial, Estructura para obtención de precios actuales, OrderSend, OrderCheck

© 2000-2025, MetaQuotes Ltd.

Estructura de resultado de solicitud comercial (MqlTradeResult)

Respondiendo a una solicitud comercial acerca de colocación de una orden en el sistema comercial, el servidor comercial devuelve los datos que contienen la información sobre el resultado de procesamiento de la solicitud comercial en forma de la estructura especial predefinida MqlTradeResult.

```
struct MqlTradeResult
  {
   uint     retcode;          // Código del resultado de operación
   ulong    deal;             // Ticket de transacción, si está concluida
   ulong    order;            // Ticket de la orden, si está colocada
   double   volume;           // Volumen de la transacción confirmado por el corredor
   double   price;            // Precio en la transacción confirmada por el corredor
   double   bid;              // Precio actual de la oferta en el mercado (precios recuota)
   double   ask;              // Precio actual de la demanda en el mercado (precios recuota)
   string   comment;          // Comentarios del corredor acerca de la operación (por defecto se rel
   uint     request_id;       // El terminal pone el identificador de la solicitud a la hora de envi
   int      retcode_external; // Código de respuesta del sistema de comercio exterior
  };
```

Descripción de campos

Campo	Descripción
retcode	Código de retorno del servidor comercial
deal	Ticket de la transacción, si está concluida. Se comunica al ejecutar la operación comercial TRADE_ACTION_DEAL
order	Ticket de la orden, si está colocada. Se comunica al ejecutar la operación comercial TRADE_ACTION_PENDING
volume	Volumen de la transacción confirmado por el corredor. Depende del tipo de ejecución de la orden
price	Precio en la transacción confirmada por el corredor. Depende del campo *deviation* en la solicitud comercial y/o del tipo de la operación comercial
bid	Precio actual de la oferta en el mercado (precios recuota)
ask	Precio actual de la demanda en el mercado (precios recuota)
comment	Comentarios del corredor acerca de la operación (por defecto se rellena con la descripción código de retorno del servidor comercial)
request_id	Identificador de solicitud puesto por el terminal al inviarla al servidor de trading
retcode_external	Código de error que es retornado por el sistema de comercio exterior. El uso y tipos de estos errores van a depender del bróker y del sistema de comercio exterior a partir del cual son enviadas las operaciones de negociación

El resultado de la operación comercial se devuelve en una variable del tipo MqlTradeResult la que se pasa como segundo parámetro a la función OrderSend() para realizar las operaciones comerciales.

© 2000-2025, MetaQuotes Ltd.

El terminal registra el identificador de la solicitud en el campo request_id a la hora de enviarla al servidor de trading por medio de las funciones OrdersSend() y OrderSendAsync(). El terminal recibe los mensajes de parte del servidor de trading sobre las transacciones comerciales realizadas y las pasa para el procesamiento a la función OnTradeTransaction() que contiene como parámetro:

- descripción de la misma transacción comercial en la estructura MqlTradeTransaction;
- descripción de la solicitud comercial que ha sido enviada desde la función OrderSend() o OrdersSendAsync(). El terminal envía el identificador de la solicitud al servidor de trading, mientras que la misma solicitud y su request_id se guardan en la memoria del terminal;
- resultado de la ejecución de la solicitud comercial en forma de la estructura MqlTradeResult donde el campo request_id contiene el identificador de esta solicitud.

La función OnTradeTransaction() obtiene tres parámetros de entrada, pero los dos últimos parámetros tiene sentido analizarlos sólo para las transacciones comerciales que tienen el tipo TRADE_TRANSACTION_REQUEST. En todos los demás casos, los datos sobre la solicitud comercial y el resultado de su ejecución no se rellenan. El ejemplo del análisis de los parámetros se muestra en el apartado Estructura de transacción comercial.

La puesta del identificador request_id para la solicitud comercial por parte del terminal a la hora de enviarla al servidor está destinada en primer lugar para el trabajo con la función asincrónica OrderSendAsync(). Este identificador permite vincular la acción ejecutada (llamada a la función OrderSend o OrderSendAsync) con el resultado de esta acción que se traspasa en OnTradeTransaction().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Envío de una solicitud comercial con el procesamiento del resultado     |
//+------------------------------------------------------------------+
bool MyOrderSend(MqlTradeRequest request,MqlTradeResult result)
  {
//--- pongamos el código del último error a cero
   ResetLastError();
//--- enviamos la solicitud
   bool success=OrderSend(request,result);
//--- si ha fallado, vamos a intentar averiguar porqué
   if(!success)
     {
      int answer=result.retcode;
      Print("TradeLog:Trade request failed. Error = ",GetLastError());
      switch(answer)
        {
         //--- recuota
         case 10004:
           {
            Print("TRADE_RETCODE_REQUOTE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- la orden no ha sido aceptada por el servidor
         case 10006:
           {
            Print("TRADE_RETCODE_REJECT");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- precio incorrecto
         case 10015:
           {
            Print("TRADE_RETCODE_INVALID_PRICE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //---  SL y/o TP incorrecto(s)
         case 10016:
           {
            Print("TRADE_RETCODE_INVALID_STOPS");
            Print("request.sl = ",request.sl," request.tp = ",request.tp);
            Print("result.ask = ",result.ask," result.bid = ",result.bid);
            break;
           }
         //--- volumen incorrecto
         case 10014:
           {
            Print("TRADE_RETCODE_INVALID_VOLUME");
            Print("request.volume = ",request.volume,"   result.volume = ",
                  result.volume);
            break;
           }
         //--- falta dinero para esta operación comercial
         case 10019:
           {
            Print("TRADE_RETCODE_NO_MONEY");
            Print("request.volume = ",request.volume,"   result.volume = ",
```

© 2000-2025, MetaQuotes Ltd.

```
                  result.volume,"   result.comment = ",result.comment);
            break;
           }
         //--- alguna otra razón, mostramos el código de respuesta del servidor
         default:
           {
            Print("Other answer = ",answer);
           }
        }
      //--- notificamos devolviendo false sobre el resultado fallido de la solicitud c
      return(false);
     }
//--- OrderSend() ha devuelto true - repetimos la respuesta
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

Estructura de transacción comercial (MqlTradeTransaction)

Como resultado de ejecución de ciertas acciones con la cuenta de trading su estado se cambia. A estas acciones les pertenecen:

- El envío de una solicitud comercial por parte de cualquier aplicación MQL5 en el terminal de cliente utilizando la función OrderSend y OrderSendAsync, con su posterior ejecución;
- El envío de una solicitud comercial a través de la interfaz gráfica del terminal y su posterior ejecución;
- El accionamiento de órdenes pendientes y órdenes Stop en el servidor;
- La ejecución de operaciones en el servidor de trading.

Como resultado de estas acciones, para la cuenta se ejecutan las transacciones comerciales:

- tramitación de la solicitud comercial;
- cambio de órdenes abiertas;
- cambio del historial de órdenes;
- cambio del historial de operaciones;
- cambio de posiciones.

Por ejemplo, al enviar una orden de compra, ésta se tramita, para la cuenta se crea una orden de compra correspondiente, se realiza la ejecución de la orden, su eliminación de la lista de las abiertas, se agrega al historial de órdenes, luego la operación correspondiente se agrega al historial, y se crea una posición nueva. Pues todas estas acciones son transacciones comerciales.

Para recibir las transacciones comerciales que se aplican a la cuenta, en MQL5 ha sido diseñado un manejador especial OnTradeTransaction(). En el primer parámetro de este manejador se traspasa la estructura MqlTradeTransaction que describe las transacciones comerciales.

```
struct MqlTradeTransaction
  {
   ulong                         deal;             // Ticket de la operación
   ulong                         order;            // Ticket de la orden
   string                        symbol;           // Nombre del instrumento financiero
   ENUM_TRADE_TRANSACTION_TYPE   type;             // Tipo de transacción comercial
   ENUM_ORDER_TYPE               order_type;       // Tipo de la orden
   ENUM_ORDER_STATE              order_state;      // Estado de la orden
   ENUM_DEAL_TYPE                deal_type;        // Tipo de la operación
   ENUM_ORDER_TYPE_TIME          time_type;        // Tipo de la orden según el tiempo de ejecución
   datetime                      time_expiration;  // Plazo de vencimiento de la orden
   double                        price;            // Precio
   double                        price_trigger;    // Precio de activación de la orden stop limitada
   double                        price_sl;         // Nivel Stop Loss
   double                        price_tp;         // Nivel Take Profit
   double                        volume;           // Volumen en lotes
   ulong                         position;         // Position ticket
   ulong                         position_by;      // Comentarios sobre la orden
  };
```

Descripción de campos

Campo	Descripción
deal	Ticket de la operación.

© 2000-2025, MetaQuotes Ltd.

Campo	Descripción
order	Ticket de la orden.
symbol	Nombre del instrumento financiero para el que se realiza la transacción.
type	Tipo de transacción comercial. El valor puede ser uno de los valores de la enumeración ENUM_TRADE_TRANSACTION_TYPE.
order_type	Tipo de orden comercial. El valor puede ser uno de los valores de la enumeración ENUM_ORDER_TYPE.
order_state	Estado de orden comercial. El valor puede ser uno de los valores de la enumeración ENUM_ORDER_STATE.
deal_type	Tipo de operación. El valor puede ser uno de los valores de la enumeración ENUM_DEAL_TYPE.
time_type	Tipo de la orden según su expiración. El valor puede ser uno de los valores de la enumeración ENUM_ORDER_TYPE_TIME.
time_expiration	Plazo de expiración de la orden pendiente (para las órdenes del tipo ORDER_TIME_SPECIFIED y ORDER_TIME_SPECIFIED_DAY).
price	Precio. En función del tipo de la transacción comercial puede ser el precio de la orden, operación o posición.
price_trigger	Precio stop (precio de activación) de la orden stop limitada (ORDER_TYPE_BUY_STOP_LIMIT y ORDER_TYPE_SELL_STOP_LIMIT).
price_sl	Precio Stop Loss. En función del tipo de la transacción comercial puede referirse al precio de la orden, operación o posición.
price_tp	Precio Take Profit. En función del tipo de la transacción comercial puede referirse al precio de la orden, operación o posición.
volume	Volumen en lotes. En función del tipo de la transacción comercial puede referirse al volumen actual de la orden, volumen de la operación o volumen de la posición.
position	Ticket de la posición a la que ha influido la transacción.
position_by	Ticket de la posición opuesta. Se usa al cerrar una posición con otra opuesta, abierta en el mismo instrumento, pero en dirección contraria.

El parámetro determinante para el análisis de una transacción que llega es su tipo que figura en el campo **type**. Por ejemplo, si la transacción es del tipo TRADE_TRANSACTION_REQUEST (el resultado de procesamiento de la solicitud por parte del servidor ha sido recibido), entonces la estructura tiene sólo un campo rellenado **type**, los demás no hace falta analizar. En este caso se puede realizar el análisis de dos campos adicionales **request** y **result** que se pasan al manejador OnTradeTransaction(), tal como se muestra en el ejemplo de abajo.

Teniendo información sobre el tipo de la operación comercial, se puede tomar la decisión sobre el análisis del estado actual de la orden, posición y transacciones (deals) en la cuenta de trading. Hay que tener en cuenta que una solicitud comercial enviada del terminal al servidor puede provocar varias transacciones (transactions) comerciales, cuya orden de llegada al terminal no se garantiza.

© 2000-2025, MetaQuotes Ltd.

La estructura MqlTradeTransaction se llena de una manera diferente en función del tipo de transacción comercial (ENUM_TRADE_TRANSACTION_TYPE):

TRADE_TRANSACTION_ORDER_* y TRADE_TRANSACTION_HISTORY_*

Para las transacciones comerciales que conciernen el procesamiento de las órdenes abiertas (TRADE_TRANSACTION_ORDER_ADD, TRADE_TRANSACTION_ORDER_UPDATE y TRADE_TRANSACTION_ORDER_DELETE) e historial de órdenes (TRADE_TRANSACTION_HISTORY_ADD, TRADE_TRANSACTION_HISTORY_UPDATE, TRADE_TRANSACTION_HISTORY_DELETE), en la estructura MqlTradeTransaction se llenan los siguientes campos:

- order - ticket de la orden;
- symbol - nombre del instrumento financiero en la orden;
- type - tipo de transacción (transaction) comercial;
- order_type - tipo de la orden;
- orders_state - estado actual de la orden;
- time_type - tipo de vencimiento de la orden;
- time_expiration - tiempo de expiración de la orden (para las órdenes con el tipo de vencimiento ORDER_TIME_SPECIFIED y ORDER_TIME_SPECIFIED_DAY);
- price - precio de la orden especificado por el cliente;
- price_trigger - precio stop de activación de la orden stop limitada (sólo para ORDER_TYPE_BUY_STOP_LIMIT y ORDER_TYPE_SELL_STOP_LIMIT);
- price_sl - precio Stop Loss de la orden (se rellena si está especificado en la orden);
- price_tp - precio Take Profit de la orden (se rellena si está especificado en la orden);
- volume - volumen actual de la orden (no ejecutado). El volumen inicial de la orden se puede conocer del historial de órdenes utilizando la función HistoryOrders*.
- position - ticket de la posición abierta, modificada o cerrada como resultado de la ejecución de una orden. Se rellena solo para las órdenes de mercado. No se rellena para TRADE_TRANSACTION_ORDER_ADD.
- position_by - ticket de la posición opuesta. Se rellena solo para las órdenes de cierre de posición con una opuesta (close by).

TRADE_TRANSACTION_DEAL_*

Para las transacciones (transactions) comerciales que conciernen el procesamiento de las operaciones (deals) (TRADE_TRANSACTION_DEAL_ADD, TRADE_TRANSACTION_DEAL_UPDATE y TRADE_TRANSACTION_DEAL_DELETE), en la estructura MqlTradeTransaction se llenan los siguientes campos:

- deal - ticket de la operación (deal);
- order - ticket de la orden a base de la cual ha sido realizada la operación (deal);
- symbol - nombre del instrumento financiero en la operación (deal);
- type - tipo de transacción (transaction) comercial;
- deal_type - tipo de la operación (deal);
- Precio — precio por el que ha sido realizada la operación (deal);
- price_sl - precio Stop Loss (se rellena si está especificado en la orden a base de la cual ha sido realizada la operación (deal));
- price_tp - precio Take Profit (se rellena si está especificado en la orden a base de la cual ha sido realizada la operación (deal));
- Volumen — volumen de la operación (deal) en lotes.

© 2000-2025, MetaQuotes Ltd.

- position - ticket de la posición abierta, modificada o cerrada como resultado de la ejecución de una operación.
- position_by - ticket de la posición opuesta. Solo se rellena para las operaciones de cierre de posición con una opuesta (out by).

TRADE_TRANSACTION_POSITION

Para las transacciones (transactions) comerciales que conciernen las modificaciones de posiciones no relacionadas con la ejecución de las operaciones (deals) (TRADE_TRANSACTION_POSITION), en la estructura MqlTradeTransaction se llenan los siguientes campos:

- symbol - nombre del instrumento financiero de la posición;
- type - tipo de transacción (transaction) comercial;
- deal_type - tipo de posición (DEAL_TYPE_BUY o DEAL_TYPE_SELL);
- price - precio medio ponderado de la apertura de la posición;
- price_sl - precio Stop Loss;
- price_tp - precio Take Profit;
- Volumen — volumen de la posición en lotes si no ha sido modificado.
- position - ticket de la posición.

> La modificación de una posición (agregación, cambio o eliminación) como resultado de ejecución de la transacción (deal) no supone la aparición tras sí la transacción (transaction) TRADE_TRANSACTION_POSITION.

TRADE_TRANSACTION_REQUEST

Para las transacciones (transactions) comerciales que describen el hecho de que la solicitud comercial haya sido procesada por el servidor y que el resultado de su procesamiento haya sido recibido (TRADE_TRANSACTION_REQUEST), en la estructura MqlTradeTransaction se rellena sólo un campo:

- type - tipo de transacción (transaction) comercial;

> Para las transacciones de este tipo hay que analizar sólo un campo - type (tipo de transacción comercial). Para obtener la información adicional hay que analizar el segundo y el tercer parámetro de la función OnTradeTransaction (request y result).

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
input int MagicNumber=1234567;

//--- activamos la clase de trading CTrade y declaramos la variable de este tipo
#include <Trade\Trade.mqh>
CTrade trade;
//--- banderas para colocación y eliminación de la orden pendiente
bool pending_done=false;
bool pending_deleted=false;
//--- aquí vamos a guardar el ticket de la orden pendiente
ulong order_ticket;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- establecemos MagicNumber para marcar todas nuestras órdenes
   trade.SetExpertMagicNumber(MagicNumber);
//--- vamos a mandar las solicitude comerciales en el modo asincrónico utilizando la f
   trade.SetAsyncMode(true);
//--- inicializamos la variable con un cero
   order_ticket=0;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- colocación de orden pendiente
   if(!pending_done)
     {
      double ask=SymbolInfoDouble(_Symbol,SYMBOL_ASK);
      double buy_stop_price=NormalizeDouble(ask+1000*_Point,(int)SymbolInfoInteger(_Sy
      bool res=trade.BuyStop(0.1,buy_stop_price,_Symbol);
      //--- si la función BuyStop() ha trabajado con éxito
      if(res)
        {
         pending_done=true;
         //--- obtenemos el resultado del envío de la solicitud desde ctrade
         MqlTradeResult trade_result;
         trade.Result(trade_result);
         //---obtenemos request_id para la solicitud enviada
         uint request_id=trade_result.request_id;
         Print("La solicitud para colocar la orden pendiente ha sido enviada. Identifi
         //--- recordamos el ticket de la orden (al usar el modo asincrónico del envío
         order_ticket=trade_result.order;
         //--- todo está hecho por eso salimos del manejador OnTick()
         return;
        }
     }
//--- eliminación de la orden pendiente
   if(!pending_deleted)
      //--- verificación adicional
      if(pending_done && (order_ticket!=0))
        {
         //--- intentamos eliminar la orden pendiente
         bool res=trade.OrderDelete(order_ticket);
         Print("OrderDelete=",res);
         //--- si la solicitud para la eliminación ha sido enviada con éxito
         if(res)
```

© 2000-2025, MetaQuotes Ltd.

```
            {
             pending_deleted=true;
             //--- obtenemos el resultado de ejecución de la solicitud
             MqlTradeResult trade_result;
             trade.Result(trade_result);
             //--- extraemos del resultado el identificador de la solicitud
             uint request_id=trade_result.request_id;
             //--- mostramos en el Diario
             Print("Se ha enviado la solicitud para la eliminación de la orden pendient
                   ". Identificador de la solicitud Request_ID=",request_id,
                   "\r\n");
             //--- registramos el ticket de la orden desde el resultado de la solicitud
             order_ticket=trade_result.order;
            }
         }
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- obtenemos el tipo de la transacción como valor de la enumeración
   ENUM_TRADE_TRANSACTION_TYPE type=(ENUM_TRADE_TRANSACTION_TYPE)trans.type;
//--- si la transacción es el resultado de procesamiento de la solicitud, mostramos só
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      Print(EnumToString(type));
      //--- mostramos la descripción de la solicitud procesada
      Print("------------RequestDescription\r\n",RequestDescription(request));
      //--- mostramos la descripción del resultado de la solicitud
      Print("------------ResultDescription\r\n",TradeResultDescription(result));
      //--- recordamos el ticket de la orden para su eliminación durante el siguiente
      if(result.order!=0)
        {
         //--- eliminamos esta orden según su ticket durante la siguiente llamada de O
         order_ticket=result.order;
         Print(" Ticket de la orden pendiente ",order_ticket,"\r\n");
        }
     }
   else // para la transacción de otro tipo mostramos la descripción completa
//--- mostramos la descripción de la transacción recibida en el Diario
      Print("------------TransactionDescription\r\n",TransactionDescription(trans));

//---
  }
//+------------------------------------------------------------------+
//|  Devuelve la descripción textual de la transacción               |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans)
  {
//---
   string desc=EnumToString(trans.type)+"\r\n";
   desc+="Symbol: "+trans.symbol+"\r\n";
   desc+="Deal ticket: "+(string)trans.deal+"\r\n";
   desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
   desc+="Order ticket: "+(string)trans.order+"\r\n";
   desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
   desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
```

© 2000-2025, MetaQuotes Ltd.

```
   desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
   desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
   desc+="Price trigger: "+StringFormat("%G",trans.price_trigger)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
   desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
//--- devolvemos la cadena obtenida
   return desc;
  }
//+------------------------------------------------------------------+
//|  Devuelve la descripción textual de la solicitud comercial       |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request)
  {
//---
   string desc=EnumToString(request.action)+"\r\n";
   desc+="Symbol: "+request.symbol+"\r\n";
   desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
   desc+="Order ticket: "+(string)request.order+"\r\n";
   desc+="Order type: "+EnumToString(request.type)+"\r\n";
   desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
   desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
   desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
   desc+="Deviation points: "+StringFormat("%G",request.deviation)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
   desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
   desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
   desc+="Comment: "+request.comment+"\r\n";
//--- devolvemos la cadena obtenida
   return desc;
  }
//+------------------------------------------------------------------+
//|  Devuelve la descripción textual del resultado de procesamiento  |
//|  de la solicitud                                                 |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result)
  {
//---
   string desc="Retcode "+(string)result.retcode+"\r\n";
   desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
   desc+="Order ticket: "+(string)result.order+"\r\n";
   desc+="Deal ticket: "+(string)result.deal+"\r\n";
   desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
   desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
   desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
   desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
   desc+="Comment: "+result.comment+"\r\n";
//--- devolvemos la cadena obtenida
   return desc;
  }
```

Véase también

Tipos de transacciones comerciales, OnTradeTransaction()

© 2000-2025, MetaQuotes Ltd.

Estructura para obtención de precios actuales (MqlTick)

Es la estructura para almacenar los últimos precios del símbolo. Sirve para recibir de una manera rápida la información más solicitada sobre los precios corrientes.

```
struct MqlTick
  {
   datetime     time;          // Hora de la última actualización de precios
   double       bid;           // Precio actual Bid
   double       ask;           // Precio actual Ask
   double       last;          // Precio actual de la última transacción (Last)
   ulong        volume;        // Volumen para el precio actual Last
   long         time_msc;      // Hora de la última actualización de los precios en mi
   uint         flags;         // Banderas de los tics
   double       volume_real;   // Volumen para el precio actual Last con precisión aur
  };
```

La variable del tipo MqlTick permite obtener los valores Ask, Bid, Last y Volume sólo con una llamada a la función SymbolInfoTick().

Todos los parámetros de cada tic siempre se llenan, independientemente de si han cambiado los datos en comparación con el tic anterior. Esto permite tener siempre el estado actual de los precios en cualquier momento, sin buscar los valores anteriores en la historia de tics. Por ejemplo, con el tic solo ha podido cambiar el precio bid, pero en la estructura, aparte del nuevo precio, se mostrarán también todos los demás parámetros: el precio anterior ask, el volumen, etcétera.

Para saber qué datos precisamente han cambiado con el tic actual, analice sus banderas:

- TICK_FLAG_BID - el tic ha cambiado el precio bid
- TICK_FLAG_ASK - el tic ha cambiado el precio ask
- TICK_FLAG_LAST - el tic ha cambiado el precio de la última transacción
- TICK_FLAG_VOLUME - el tic ha cambiado el volumen
- TICK_FLAG_BUY - el tic ha aparecido como resultado de una transacción de compra
- TICK_FLAG_SELL - el tic ha aparecido como resultado de una transacción de venta

Ejemplo:

```
void OnTick()
  {
   MqlTick last_tick;
//---
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      Print(last_tick.time,": Bid = ",last_tick.bid,
            " Ask = ",last_tick.ask,"  Volume = ",last_tick.volume);
     }
   else Print("SymbolInfoTick() failed, error = ",GetLastError());
//---
  }
```

Véase también

© 2000-2025, MetaQuotes Ltd.

Estructuras y clases, CopyTicks(), SymbolInfoTick()

© 2000-2025, MetaQuotes Ltd.

Estructuras del calendario económico

En este apartado se describen las estructuras para trabajar con el Calendario Económico, disponible directamente en la plataforma MetaTrader. El calendario económico es una enciclopedia ya preparada que contiene las descripciones de los más importantes indicadores macroeconómicos, sus fechas de salida y su nivel de importancia. Los valores actuales de los indicadores macroeconímicos llegan a la plataforma MetaTrader justo tras su publicación, y se representan en un gráfico en forma de rótulos: esto permitirá monitorear visualmente los índices necesarios en función del país, la divisa y la importancia.

Las funciones del Calendario Económico permiten analizar de forma automática los eventos entrantes según nuestros propios criterios de importancia, y también en función de los países/divisas requeridos.

Las descripciones de los países de indican con la estructura MqlCalendarCountry. Se usa en las funciones CalendarCountryById() y CalendarCountries()

```
struct MqlCalendarCountry
  {
   ulong                id;                // identificador del paí
   string               name;              // nombre de texto del p
   string               code;              // nombre en clave del p
   string               currency;          // código de la divisa d
   string               currency_symbol;   // símbolo/signo de la d
   string               url_name;          // nombre del país usado
  };
```

Las descripciones de los eventos se indican con la estructura MqlCalendarEvent. Se usa en las funciones CalendarEventById(), CalendarEventByCountry() y CalendarEventByCurrency()

```
struct MqlCalendarEvent
  {
   ulong                id;                // identificador del eve
   ENUM_CALENDAR_EVENT_TYPE      name;     // tipo de evento de la
   ENUM_CALENDAR_EVENT_SECTOR    sector;   // sector con el que se
   ENUM_CALENDAR_EVENT_FREQUENCY frequency;// frecuencia (periodici
   ENUM_CALENDAR_EVENT_TIMEMODE  sector;   // modo temporal del eve
   ulong                country_id;        // identificador del paí
   ENUM_CALENDAR_EVENT_UNIT      unit;     // unidad de medición de
   ENUM_CALENDAR_EVENT_IMPORTANCE importance;// importancia del event
   ENUM_CALENDAR_EVENT_MULTIPLIER multiplier;// multiplicador del val
   uint                 digits;            // número de decimales t
   string               source_url;        // URL de la fuente dond
   string               event_code;        // código del evento
   string               name;              // nombre de texto del e
  };
```

© 2000-2025, MetaQuotes Ltd.

Los valores de los eventos se indican con la estructura MqlCalendarValue. Se usa en las funciones CalendarValueById(), CalendarValueHistoryByEvent(), CalendarValueHistory(), CalendarValueLastByEvent() y CalendarValueLast()

```
struct MqlCalendarValue
  {
   ulong                             id;                // ID del valor
   ulong                             event_id;          // ID del evento
   datetime                          time;              // hora y fecha evento
   datetime                          period;            // período de informe de
   int                               revision;          // revisión del indicado
   long                              actual_value;      // valor real en ppm o I
   long                              prev_value;        // valor anterior en ppm
   long                              revised_prev_value; // valor anterior revisa
   long                              forecast_value;    // valor previsto en ppm
   ENUM_CALENDAR_EVENT_IMPACT        impact_type;       // impacto potencial en
  //--- funciones para comprobar valores
   bool                         HasActualValue(void) const;   // devuelve true si en e
   bool                         HasPreviousValue(void) const; // devuelve true si en e
   bool                         HasRevisedValue(void) const;  // devuelve true si en e
   bool                         HasForecastValue(void) const; // devuelve true si en e
  //--- funciones para obtener valores
   double                       GetActualValue(void) const;   // devuelve actual_value
   double                       GetPreviousValue(void) const; // devuelve prev_value o
   double                       GetRevisedValue(void) const;  // devuelve revised_prev
   double                       GetForecastValue(void) const; // devuelve forecast_val
  };
```

La estructura MqlCalendarValue proporciona un método para obtener e comprobar valores a partir de los campos *actual_value*, *forecast_value*, *prev_value* y *revised_prev_value*. Si el valor del campo no está definido, éste almacenará el valor LONG_MIN (-9223372036854775808).

Es necesario tener en cuenta que los valores de estos campos se almacenan multiplicados por un factor de un millón. Eso quiere decir que cuando las funciones CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent y CalendarValueLast reciben valores en MqlCalendarValue, es necesario comprobar que los valores LONG_MIN de esos campos son idénticos; y si el valor está definido en el campo, para obtener el valor, el valor del campo debe dividirse entre 1000 000 (un millón). Otra forma de obtener valores es verificar y obtener valores mediante funciones de la propia estructura MqlCalendarValue.

Ejemplo de procesamiento de eventos de calendario:

```
//--- creamos nuestra propia estructura para almacenar eventos de calendario con valor
struct AdjustedCalendarValue
  {
   ulong                             id;                // ID del valor
   ulong                             event_id;          // ID del evento
   datetime                          time;              // hora y fecha evento
   datetime                          period;            // período de informe de
   int                               revision;          // revisión del indicado
```

© 2000-2025, MetaQuotes Ltd.

```
   double                            actual_value;          // valor actual do indic
   double                            prev_value;            // valor anterior del in
   double                            revised_prev_value;    // valor anterior revisa
   double                            forecast_value;        // valor previsto del in
   ENUM_CALENDAR_EVENT_IMPACT        impact_type;           // impacto potencial en
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
//--- código de país aplicado a la Unión Europea según la norma ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- obtenemos todos los valores de eventos para la Unión Europea
   MqlCalendarValue values[];
//--- definimos los límites del rango del que tomamos los eventos
   datetime date_from=D'01.01.2021';  // tomamos todos os eventos a partir de 2021
   datetime date_to=0;                // 0 representa todos los eventos conocidos, inc
//--- solicitamos el historial de eventos de la Unión Europea a partir de 2021
   if(!CalendarValueHistory(values, date_from, date_to, EU_code))
     {
      PrintFormat("Erro! No se pudieron obtener los eventos por país país country_code
      PrintFormat("Código de error: %d", GetLastError());
      return;
     }
   else
      PrintFormat("Se recibieron valores de eventos para el país country_code=%s: %d",
                  EU_code, ArraySize(values));
//--- reducimos el tamaño de la matriz para mostrarla en el Log
   if(ArraySize(values)>5)
      ArrayResize(values, 5);
//--- mostramos los valores de los eventos en el Log tal como están, sin verificacione
   Print("Mostramos los valores del calendario tal cual");
   ArrayPrint(values);

//--- comprobamos los valores del campo y los actualizamos a los valores actuales
//--- primera opción para comprobar y obtener valores
   AdjustedCalendarValue values_adjusted_1[];
   int total=ArraySize(values);
   ArrayResize(values_adjusted_1, total);
//--- copiamos los valores verificados y corregidos
   for(int i=0; i<total; i++)
     {
      values_adjusted_1[i].id=values[i].id;
      values_adjusted_1[i].event_id=values[i].event_id;
      values_adjusted_1[i].time=values[i].time;
      values_adjusted_1[i].period=values[i].period;
      values_adjusted_1[i].revision=values[i].revision;
```

© 2000-2025, MetaQuotes Ltd.

```
      values_adjusted_1[i].impact_type=values[i].impact_type;
      //--- realizamos la verificación de valores y dividimos por 1000 000
      if(values[i].actual_value==LONG_MIN)
         values_adjusted_1[i].actual_value=double("nan");
      else
         values_adjusted_1[i].actual_value=values[i].actual_value/1000000.;

      if(values[i].prev_value==LONG_MIN)
         values_adjusted_1[i].prev_value=double("nan");
      else
         values_adjusted_1[i].prev_value=values[i].prev_value/1000000.;

      if(values[i].revised_prev_value==LONG_MIN)
         values_adjusted_1[i].revised_prev_value=double("nan");
      else
         values_adjusted_1[i].revised_prev_value=values[i].revised_prev_value/1000000.

      if(values[i].forecast_value==LONG_MIN)
         values_adjusted_1[i].forecast_value=double("nan");
      else
         values_adjusted_1[i].forecast_value=values[i].forecast_value/1000000.;
     }
   Print("Primera forma de comprobar y obtener valores de calendario");
   ArrayPrint(values_adjusted_1);

//--- segunda opción para comprobar y obtener valores
   AdjustedCalendarValue values_adjusted_2[];
   ArrayResize(values_adjusted_2, total);
//--- copiamos los valores verificados y corregidos
   for(int i=0; i<total; i++)
     {
      values_adjusted_2[i].id=values[i].id;
      values_adjusted_2[i].event_id=values[i].event_id;
      values_adjusted_2[i].time=values[i].time;
      values_adjusted_2[i].period=values[i].period;
      values_adjusted_2[i].revision=values[i].revision;
      values_adjusted_2[i].impact_type=values[i].impact_type;
      //--- realizamos las comprobaciones y obtenemos valores
      if(values[i].HasActualValue())
         values_adjusted_2[i].actual_value=values[i].GetActualValue();
      else
         values_adjusted_2[i].actual_value=double("nan");

      if(values[i].HasPreviousValue())
         values_adjusted_2[i].prev_value=values[i].GetPreviousValue();
      else
         values_adjusted_2[i].prev_value=double("nan");

      if(values[i].HasRevisedValue())
```

© 2000-2025, MetaQuotes Ltd.

```
         values_adjusted_2[i].revised_prev_value=values[i].GetRevisedValue();
      else
         values_adjusted_2[i].revised_prev_value=double("nan");

      if(values[i].HasForecastValue())
         values_adjusted_2[i].forecast_value=values[i].GetForecastValue();
      else
         values_adjusted_2[i].forecast_value=double("nan");
     }
   Print("Segunda forma de comprobar y obtener valores de calendario");
   ArrayPrint(values_adjusted_2);

//--- tercera opción para verificar y obtener valores
   AdjustedCalendarValue values_adjusted_3[];
   ArrayResize(values_adjusted_3, total);
//--- copiamos los valores verificados y corregidos
   for(int i=0; i<total; i++)
     {
      values_adjusted_3[i].id=values[i].id;
      values_adjusted_3[i].event_id=values[i].event_id;
      values_adjusted_3[i].time=values[i].time;
      values_adjusted_3[i].period=values[i].period;
      values_adjusted_3[i].revision=values[i].revision;
      values_adjusted_3[i].impact_type=values[i].impact_type;
      //--- obtenemos valores sin verificaciones
      values_adjusted_3[i].actual_value=values[i].GetActualValue();
      values_adjusted_3[i].prev_value=values[i].GetPreviousValue();
      values_adjusted_3[i].revised_prev_value=values[i].GetRevisedValue();
      values_adjusted_3[i].forecast_value=values[i].GetForecastValue();
     }
   Print("Tercera forma de obtener valores de calendario, sin verificaciones");
   ArrayPrint(values_adjusted_3);
  }
/*
   Se reciben los valores de eventos según country_code=EU: 1051
  Mostramos los valores del calendario como son
         [id] [event_id]              [time]            [period] [revision]        [act
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0 -922337203
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0
  Primera forma de verificar y obtener valores de calendario
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
```

© 2000-2025, MetaQuotes Ltd.

```
  Segunda forma de comprobar y obtener valores de calendario
        [id] [event_id]              [time]            [period] [revision] [actual_va
  [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
  [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
  [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
  [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
  [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
  Tercera forma de obtener valores de calendario, sin verificaciones
        [id] [event_id]              [time]            [period] [revision] [actual_va
  [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
  [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
  [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
  [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
  [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
*/
```

La frecuencia (periodicidad) del evento se muestra en la estructura MqlCalendarEvent. Los posibles valores se indican en la numeración **ENUM_CALENDAR_EVENT_FREQUENCY**

Identificador	Descripción
CALENDAR_FREQUENCY_NONE	La frecuencia de la publicación no ha sido indicada
CALENDAR_FREQUENCY_WEEK	Publicación semanal
CALENDAR_FREQUENCY_MONTH	Publicación mensual
CALENDAR_FREQUENCY_QUARTER	Publicación trimestral
CALENDAR_FREQUENCY_YEAR	Publicación anual
CALENDAR_FREQUENCY_DAY	Publicación diaria

El tipo de evento se indica en la estructura MqlCalendarEvent. Los posibles valores se indican en la numeración **ENUM_CALENDAR_EVENT_TYPE**

Identificador	Descripción
CALENDAR_TYPE_EVENT	Evento (reunión, discurso, etcétera)
CALENDAR_TYPE_INDICATOR	Indicador
CALENDAR_TYPE_HOLIDAY	Festivo

El sector de la economía con el que se relaciona el evento se indica en la estructura MqlCalendarEvent. Los posibles valores se indican en la numeración **ENUM_CALENDAR_EVENT_SECTOR**

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
CALENDAR_SECTOR_NONE	El sector no ha sido indicado
CALENDAR_SECTOR_MARKET	Mercado, bolsa
CALENDAR_SECTOR_GDP	Producto interior bruto (PIB)
CALENDAR_SECTOR_JOBS	Mercado laboral
CALENDAR_SECTOR_PRICES	Precios
CALENDAR_SECTOR_MONEY	Dinero
CALENDAR_SECTOR_TRADE	Comercio
CALENDAR_SECTOR_GOVERNMENT	Gobierno
CALENDAR_SECTOR_BUSINESS	Negocios
CALENDAR_SECTOR_CONSUMER	Consumo
CALENDAR_SECTOR_HOUSING	Vivienda
CALENDAR_SECTOR_TAXES	Impuestos
CALENDAR_SECTOR_HOLIDAYS	Festivos

La importancia del evento se indica en la estructura MqlCalendarEvent. Los posibles valores se indican en la numeración **ENUM_CALENDAR_EVENT_IMPORTANCE**

Identificador	Descripción
CALENDAR_IMPORTANCE_NONE	El nivel de importancia no ha sido indicado
CALENDAR_IMPORTANCE_LOW	Baja importancia
CALENDAR_IMPORTANCE_MODERATE	Media importancia
CALENDAR_IMPORTANCE_HIGH	Alta importancia

El tipo de la unidad de medición en el que se dan los valores del evento se indica en la estructura MqlCalendarEvent. Los posibles valores se indican en la numeración **ENUM_CALENDAR_EVENT_UNIT**

Identificador	Descripción
CALENDAR_UNIT_NONE	La unidad de medición no ha sido indicada
CALENDAR_UNIT_PERCENT	Porcentaje
CALENDAR_UNIT_CURRENCY	Divisa nacional
CALENDAR_UNIT_HOUR	Número de horas
CALENDAR_UNIT_JOB	Número de puestos de trabajo

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
CALENDAR_UNIT_RIG	Plataformas petrolíferas
CALENDAR_UNIT_USD	Dólares USA
CALENDAR_UNIT_PEOPLE	Número de personas
CALENDAR_UNIT_MORTGAGE	Número de créditos hipotecarios
CALENDAR_UNIT_VOTE	Número de votos
CALENDAR_UNIT_BARREL	Número de barriles
CALENDAR_UNIT_CUBICFEET	Número de pies cúbicos
CALENDAR_UNIT_POSITION	Número de puestos de trabajo
CALENDAR_UNIT_BUILDING	Número de construcciones

En ciertos casos, los valores del indicador económico necesitan que se indique el multiplicador especificado en la estructura MqlCalendarEvent. Los posibles valores se indican en la numeración **ENUM_CALENDAR_EVENT_MULTIPLIER**

Identificador	Descripción
CALENDAR_MULTIPLIER_NONE	El multiplicador no ha sido indicado
CALENDAR_MULTIPLIER_THOUSANDS	Miles
CALENDAR_MULTIPLIER_MILLIONS	Millones
CALENDAR_MULTIPLIER_BILLIONS	Miles de millones
CALENDAR_MULTIPLIER_TRILLIONS	Billones

La influencia potencial de un evento en el curso de la divisa nacional se indica en la estructura MqlCalendarValue. Los posibles valores se indican en la numeración **ENUM_CALENDAR_EVENT_IMPACT**

Identificador	Descripción
CALENDAR_IMPACT_NA	La influencia no ha sido indicada
CALENDAR_IMPACT_POSITIVE	Influencia positiva
CALENDAR_IMPACT_NEGATIVE	Influencia negativa

La hora del evento se indica en la estructura MqlCalendarEvent. Los posibles valores se indican en la numeración **ENUM_CALENDAR_EVENT_TIMEMODE**

Identificador	Descripción
CALENDAR_TIMEMODE_DATETIME	La fuente publica la hora exacta del evento

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
CALENDAR_TIMEMODE_DATE	El evento ocupa el día completo
CALENDAR_TIMEMODE_NOTIME	La fuente no publica la hora del evento
CALENDAR_TIMEMODE_TENTATIVE	La fuente no publica de antemano la hora exacta del evento, solo el día. La hora se concreta cuando tiene lugar el evento

Ver también

Calendario económico

© 2000-2025, MetaQuotes Ltd.

Códigos de errores y advertencias

Este apartado contiene las siguientes descripciones:

- Códigos de retorno del servidor comercial – análisis de resultados del envío de una solicitud comercial mandada por la función OrderSend();
- Advertencias del compilador – códigos de los mensajes de advertencia mostrados durante la compilación (nos son errores);
- Errores de compilación – códigos de los mensajes de error en caso del intento fallido de compilación;
- Errores de tiempo de ejecución – códigos de errores durante la ejecución de un programa mql5, los que se puede obtener utilizando la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

Códigos de retorno del servidor comercial

Todas las órdenes respecto a la ejecución de las operaciones comerciales se mandan en forma de una estructura de solicitudes comerciales MqlTradeRequest a través de la función OrderSend(). El resultado de ejecución de esta función se coloca en la estructura MqlTradeResult, su campo *retcode* contiene el código de retorno del servidor comercial.

Código	Identificador	Descripción
10004	TRADE_RETCODE_REQUOTE	Recuota
10006	TRADE_RETCODE_REJECT	Solicitud rechazada
10007	TRADE_RETCODE_CANCEL	Solicitud cancelada por el agente
10008	TRADE_RETCODE_PLACED	Orden colocada
10009	TRADE_RETCODE_DONE	Solicitud ejecutada
10010	TRADE_RETCODE_DONE_PARTIAL	Solicitud ejecutada parcialmente
10011	TRADE_RETCODE_ERROR	Error al procesar la solicitud
10012	TRADE_RETCODE_TIMEOUT	Solicitud cancelada al expirar el plazo
10013	TRADE_RETCODE_INVALID	Solicitud no válida
10014	TRADE_RETCODE_INVALID_VOLUME	Volumen en la solicitud no válido
10015	TRADE_RETCODE_INVALID_PRICE	Precio en la solicitud no válido
10016	TRADE_RETCODE_INVALID_STOPS	Stops en la solicitud no válido
10017	TRADE_RETCODE_TRADE_DISABLED	Transacciones comerciales están prohibidas
10018	TRADE_RETCODE_MARKET_CLOSED	Mercado está cerrado
10019	TRADE_RETCODE_NO_MONEY	Falta de medios monetarios para cumplir la solicitud
10020	TRADE_RETCODE_PRICE_CHANGED	Precios se han cambiado
10021	TRADE_RETCODE_PRICE_OFF	Faltan las cotizaciones para procesar la solicitud
10022	TRADE_RETCODE_INVALID_EXPIRATION	Fecha de expiración no válida de la orden en la solicitud
10023	TRADE_RETCODE_ORDER_CHANGED	Estado de la orden se ha cambiado
10024	TRADE_RETCODE_TOO_MANY_REQUESTS	Solicitudes muy frecuentes
10025	TRADE_RETCODE_NO_CHANGES	Sin cambios en la solicitud
10026	TRADE_RETCODE_SERVER_DISABLES_AT	Autotrading está prohibido por el servidor

© 2000-2025, MetaQuotes Ltd.

Código	Identificador	Descripción
10027	TRADE_RETCODE_CLIENT_DISABLES_AT	Autotrading está prohibido por el terminal de cliente
10028	TRADE_RETCODE_LOCKED	Solicitud está bloqueada para procesar
10029	TRADE_RETCODE_FROZEN	Orden o posición están congeladas
10030	TRADE_RETCODE_INVALID_FILL	Está especificado el tipo de ejecución de orden no válido
10031	TRADE_RETCODE_CONNECTION	No hay conexión con el servidor de comercio
10032	TRADE_RETCODE_ONLY_REAL	Operación permitida únicamente para las cuentas reales
10033	TRADE_RETCODE_LIMIT_ORDERS	Alcanzado el límite del número de órdenes pendientes
10034	TRADE_RETCODE_LIMIT_VOLUME	Alcanzado el límite del volumen de órdenes y posiciones para este símbolo
10035	TRADE_RETCODE_INVALID_ORDER	Tipo de orden inválido o prohibido
10036	TRADE_RETCODE_POSITION_CLOSED	Posición con el POSITION_IDENTIFIER especificado ya está cerrada
10038	TRADE_RETCODE_INVALID_CLOSE_VOLUME	El volumen cerrado supera el volumen actual de la posición
10039	TRADE_RETCODE_CLOSE_ORDER_EXIST	Para la posición indicada ya existe una orden de cierre. Puede surgir al trabajar en el sistema de cobertura: • al intentar cerrar una posición con una opuesta, si ya existe una orden de cierre de esta posición • al intentar realizar un cierre total o parcial, si el volumen sumado de las órdenes de cierre ya existentes y la orden colocada de nuevo supera el volumen actual de la posición
10040	TRADE_RETCODE_LIMIT_POSITIONS	El número de posiciones abiertas que puede existir en la cuenta de forma simultánea puede verse limitado por los ajustes del servidor. Una vez alcanzado el límite, el servidor retornará el error TRADE_RETCODE_LIMIT_POSITIONS como respuesta a la colocación de una orden. Esta restricción actúa de forma diferente, dependiendo del tipo de registro de posición usado en la cuenta: • Sistema de compensación — tiene en cuenta el número de posiciones

© 2000-2025, MetaQuotes Ltd.

Código	Identificador	Descripción
		abiertas. Al alcanzar el límite, la plataforma no permite establecer nuevas órdenes cuya ejecución pueda provocar el aumento de posiciones abiertas. De hecho, la plataforma permite colocar órdenes solo de aquellos símbolos de los que ya existen posiciones abiertas. En el sistema de compensación, al comprobar el límite, no se tienen en cuenta las órdenes pendientes actuales, puesto que su ejecución puede causar cambios en las posiciones actuales, pero no el aumento de su número. • Sistema de cobertura — aparte de las posiciones abiertas, se tienen en consideración las órdenes pendientes colocadas, puesto que su activación siempre conduce a la apertura de una nueva posición. Al alcanzar el límite, la plataforma no permite colocar órdenes de mercado para la apertura de posiciones, así como órdenes pendientes.
10041	TRADE_RETCODE_REJECT_CANCEL	La solicitud de activación de la orden pendiente ha sido rechazada, la orden ha sido cancelada
10042	TRADE_RETCODE_LONG_ONLY	La solicitud ha sido rechazada, ya que para el símbolo se ha establecido la regla "**Solo se permiten posiciones largas**" (POSITION_TYPE_BUY)
10043	TRADE_RETCODE_SHORT_ONLY	La solicitud ha sido rechazada, ya que para el símbolo se ha establecido la regla "**Solo se permiten posiciones cortas**" (POSITION_TYPE_SELL)
10044	TRADE_RETCODE_CLOSE_ONLY	La solicitud ha sido rechazada, ya que para el símbolo se ha establecido la regla "**Solo se permite cerrar las posiciones existentes**"
10045	TRADE_RETCODE_FIFO_CLOSE	La solicitud ha sido rechazada, ya que para el símbolo se ha establecido la regla "**Solo se permite cerrar las posiciones existentes según el principio FIFO**" (ACCOUNT_FIFO_CLOSE=true)

© 2000-2025, MetaQuotes Ltd.

Código	Identificador	Descripción
10046	TRADE_RETCODE_HEDGE_PROHIBITED	La solicitud ha sido rechazada, ya que para la cuenta comercial se ha establecido la regla "**Prohibido cerrar posiciones opuestas de un mismo símbolo**". Por ejemplo, si en la cuenta tenemos una posición Buy, el usuario no podrá abrir una posición Sell o colocar una orden pendiente de venta. La regla se aplica solo en las cuentas con sistema de cobertura de registro de posiciones (ACCOUNT_MARGIN_MODE=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING).

© 2000-2025, MetaQuotes Ltd.

Advertencias del compilador

Las advertencias del compilador tienen un carácter informativo, no son mensajes de error.

Número	Descripción
21	Escritura incompleta de fecha en la cadena datetime
22	Números erróneos en la cadena datetime para la fecha, se requiere: año 1970<=X<=3000 mes 0<X<=12 día 0<X<= 31/30/28(29)....
23	Números erróneos en la cadena datetime para la hora, se requiere: hora 0<=X<24 minuto 0<=X<60
24	Color incorrecto en el formato RGB: uno de los componentes RGB es menos de 0 o más de 255
25	Caracter desconocido en la secuencia de escape. Se conocen: \n \r \t \\ \" \' \X \x
26	Volumen de variables locales muy grande (>512Kb) de la función, reduzca la cantidad
29	Enumeración ya está definida (duplicación) - miembros serán añadidos a la primera definición
30	Sobrescritura de una macro
31	La variable está declarada pero no se utiliza en ningún sitio
32	Constructor tiene que ser del tipo void
33	Destructor tiene que ser del tipo void
34	Constante no entra en el rango de los enteros (X>_UI64_MAX || X<_I64_MIN) y será transformada en el tipo double
35	HEX muy largo, más de 16 caracteres significativos (se cortan los medio bytes "nibbles" mayores)
36	No hay ni un medio byte en la HEX cadena "0x"
37	No hay funciones - nada para procesar
38	Se usa una variable no inicializada
41	Función sin cuerpo, tampoco se llama
43	Posibles pérdidas de datos durante la conversión del tipo. Ejemplo: int x=(double)z;
44	Pérdida de precisión (datos) durante la conversión de una constante. Ejemplo: int x=M_PI
45	En las operaciones de comparación hay diferencia entre los signos de operandos. Ejemplo: (char)c1>(uchar)c2

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
46	Problemas con importación de la función - se requiere la declaración de #import o la importación de funciones ya está cerrada
47	Descripción es muy larga - los caracteres sobrantes no serán incluidos en el archivo ejecutable
48	Cantidad de buffers de indicadores declarados es menor de la requerida
49	Color para dibujar una serie gráfica en el indicador no está especificado
50	No hay series gráficas para dibujar el indicador
51	Función manejadora "OnStart" no ha sido encontrada en el script
52	Función manejadora "OnStart" está definida con parámetros erróneos
53	Función "OnStart" puede ser definida sólo en un script
54	Función "OnInit" está definida con parámetros erróneos
55	Función "OnInit" no se usa en los scripts
56	Función 'OnDeinit' está definida con parámetros erróneos
57	Función 'OnDeinit' no se usa en los scripts
58	Hay dos funciones 'OnCalculate' definidas. Se usará la función OnCalculate() en un array de precios
59	Detectado el sobrante a la hora de calcular una constante de números enteros
60	Probablemente la variable no esté inicializada.
61	Esta declaración hace que sea imposible referirse a la variable local declarada en la línea especificada
62	Esta declaración hace que sea imposible referirse a la variable global declarada en la línea especificada
63	No se puede usar para los arrays estáticos
64	Esta declaración hace imposible la referencia a la variable predefinida
65	Valor de expresión siempre es true/false
66	El uso de una variable o expresión del tipo bool en las operaciones matemáticas puede resultar inseguro
67	El resultado de aplicación del operador menos unario al tipo sin signo ulong es indefinido
68	La versión especificada en la propiedad #property version es inadmisible para la colocación en el apartado Tienda, el formato correcto sería #property version "XXX.YYY"
69	Falta de la expresión para la ejecución según la condición

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
70	Tipo de función que se devuelve es incorrecto, o parámetros incorrectos durante la declaración de la función-manejadora del evento
71	Se requiere la conversión de estructuras explícita al mismo tipo
72	Esta declaración hace imposible el acceso directo al miembro de una clase que ha sido declarado en esta cadena. El acceso será posible sólo con el uso de la operación de resolución de contexto ::
73	La constante binaria es muy grande, los dígitos superiores serán truncados
74	El parámetro en el método de la clase heredada se diferencia con el modificador const, la función derivada ha reiniciado la función del padre
75	Desviación negativa o con un valor demasiado grande en la operación de la desviación a nivel de bits, el resultado de la ejecución no está determinado
76	La función tiene que devolver el valor
77	La función del tipo void no debe devolver el valor
78	No todas las variantes de la ejecución devuelven el valor
79	Las expresiones a nivel global no están permitidas
80	Es posible un error en la secuencia de ejecución de las operaciones, utilice los paréntesis para la especificación explícita del orden
81	Se ha encontrado dos tipos de la llamada a OnCalCulate(). Se llamará la versión con el uso de la serie temporal OHLC
82	La estructura no contiene miembros, el tamaño será igualado a 1 byte
83	No hay procesamiento del resultado de la ejecución de la función
84	El indicador incluido como recurso ha sido compilado en modo de depuración. Eso reduce su rendimiento. Hay que recompilarlo para subir la velocidad de su funcionamiento
85	Hay un código demasiado grande en la cadena, tiene que encontrarse dentro del rango de 0 a 65535
86	Hay un carácter de servicio no reconocido en la cadena
87	La propiedad del indicador (que visualiza en la ventana principal o ventana separada) no está especificada. Se aplicará la propiedad #property indicator_chart_window

© 2000-2025, MetaQuotes Ltd.

Errores de compilación

MetaEdtior 5 (editor de programas mql5) muestra mensajes sobre los errores del programa que han sido detectados por el compilador built-in durante el proceso de compilación. La lista de estos errores viene en la tabla de abajo. Para compilar el código fuente en un código ejecutable pulse **F7**. Los programas que tienen errores no podrán ser compilados hasta que los errores especificados por el compilador no sean corregidos.

Número	Descripción
100	Error de lectura de archivo
101	Error de apertura de un archivo *.EX5 con el fin de escribirlo para guardar
103	Memoria insuficiente para terminar la compilación
104	Unidad sintáctica vacía no reconocida por el compilador
105	Nombre del archivo incorrecto en #include
106	Error de acceso a un archivo en #include (tal vez el archivo no exista)
108	Nombre inapropiado para #define
109	Comando desconocido del preprocesador (estos valen #include,#define,#property,#import)
110	Símbolo desconocido para el compilador
111	Función sin implementar (hay descripción, pero no hay cuerpo)
112	Falta comilla doble (")
113	Falta paréntesis angular izquierdo (<) o comilla doble (")
114	Falta comilla simple (')
115	Falta paréntesis angular derecho ">"
116	En declaración no está especificado el tipo
117	No hay operador de retorno return, o hay pero no en todas las ramas de ejecución
118	Se esperaba el corchete que abre del parámetro de la llamada
119	Error de escritura EX5
120	Acceso incorrecto a los elementos de un array
121	Función no tiene el tipo void y el operador return debe devover un valor
122	Declaración del destructor incorrecta
123	Faltan dos puntos ":"
124	Variable ya está declarada
125	Variable con este identificador ya está declarada
126	Nombre de variable muy largo (>250 caracteres)

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
127	Estructura con este identificador ya está definida
128	Estructura no definida
129	Miembro de estructura con este nombre ya está definido
130	No existe este miembro de estructura
131	Corchetes no hacen pareja
132	Se espera el paréntesis izquierdo "("
133	Llaves sin hacer pareja (falta "}")
134	Complicado para la compilación (muchísimas ramas, la pila entera de niveles rellena)
135	Error de apertura de un archivo para la lectura
136	Falta memoria para cargar archivo de origen en la memoria
137	Se espera una variable
138	Referencia no puede ser inicializada
140	Se esperaba asignación (surge con la declaración)
141	Se espera la llave izquierda "{"
142	Sólo un array dinámico puede figurar como parámetro
143	Uso del tipo "void" es inadmisible
144	No hay par para ")" o "]", es decir, falta "(" o "["
145	No hay par para "(" o "[", es decir, falta ")" o "]"
146	Tamaño del array incorrecto
147	Demasiados parámetros (>64)
149	Este token no se espera aquí
150	Uso de operación inadmisible (operandos incorrectos)
151	Expresión del tipo void es inadmisible
152	Se espera operador
153	Uso incorrecto de break
154	Se espera punto y coma ";"
155	Se espera coma ","
156	Tiene que ser un tipo de clase, y no de estructura
157	Se esperaba una expresión

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
158	En HEX hay "un símbolo que no es HEX" o número demasiado largo (número de dígitos > 511)
159	Cadena-constante tiene más de 65534 símbolos
160	Definición de una función aquí es inaceptable
161	Fin de programa inesperado
162	Declaración adelantada está prohibida para las estructuras
163	Función con este nombre ya está definida y tiene otro tipo del valor de retorno
164	Función con este nombre ya está definida y tiene otro conjunto de parámetros
165	Función con este nombre ya está definida y implementada
166	No se ha encontrado la sobrecarga para esta función
167	Función con valor devuelto del tipo void no puede devolver valor
168	Función no definida
170	Se espera un valor
171	En la expresión case se aceptan sólo las constantes de números enteros
172	Valor para case en este switch ya ha sido usado
173	Se espera el valor de números enteros
174	En la expresión #import se espera el nombre de archivo
175	Expresiones a nivel global no están permitidas
176	Falta paréntesis ")" antes de ";"
177	A la izquierda del signo igualdad se supone una variable
178	Resultado de la expresión no se usa
179	Declaración de las variables en case no se permite
180	Conversión implícita de una cadena a un número
181	Conversión implícita de un número a una cadena
182	Llamada ambigua a una función sobrecargada (valen varias sobrecargas)
183	Inadmisible else sin correspondiente if
184	Inadmisible case o default sin correspondiente switch
185	Uso inadmisible de elipse
186	La sucesión inicializadora tiene más elementos que la variable inicializada
187	Se espera una constante para case
188	Se requiere una expresión constante

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
189	Una variable constante no puede ser cambiada
190	Se espera una paréntesis o coma (declaración del miembro del array)
191	Identificador de la enumeración ya se utiliza
192	Enumeración no puede tener modificadores de acceso (const, extern, static)
193	Miembro de enumeración ya ha sido declarado con otro valor
194	Existe una variable definida con el mismo nombre
195	Existe una estructura definida con el mismo nombre
196	Se espera el nombre del miembro de enumeración
197	Se espera una expresión de números enteros
198	División por cero en una expresión constante
199	Número de parámetros incorrecto en la función
200	Parámetro por referencia tiene que ser una variable
201	Se espera una variable del mismo tipo para pasarla por referencia
202	Una variable constante no puede ser pasada por referencia no constante
203	Se requiere una constante positiva de números enteros
204	Error de acceso a un miembro de clase protegido
205	Importación ya está definida por otro medio
208	Archivo ejecutable no está creado
209	Punto de entrada 'OnCalculate' para el indicador no ha sido encontrado
210	Operador continue puede ser usado sólo dentro del ciclo
211	Error de acceso al miembro de clase private (cerrado)
213	Método de estructura o clase no está declarado
214	Error de acceso al método de clase private (cerrado)
216	No se permite copiar las estructuras con objetos
218	Índice sale del rango de array
219	No se permite la inicialización de arrays en la declaración de métodos o clases
220	Constructor de clase no puede tener parámetro
221	Destructor de clase no puede tener parámetro
222	Ya está declarado método de clase o estructura con este nombre y parámetros
223	Se espera un operando

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
224	Ya existe método de clase o estructura con este nombre pero con otros parámetros (declaración!=implementación)
225	Función importada sin describir
226	La función ZeroMemory() no está permitida para las clases con miembros protegidos o herencia
227	Llamada ambigua a una función sobrecargada (coincidencia exacta de parámetros para varias sobrecargas)
228	Se espera el nombre de variable
229	No se puede declarar una referencia en este sitio
230	Ya se usa como el nombre de enumeración
232	Se espera clase o estructura
235	No se puede llamar a delete para eliminar el array
236	Se espera el operador ' while '
237	El operador delete tiene que tener un puntero
238	Ya hay default para este switch
239	Error sintáctico
240	Sucesión escape puede encontrarse sólo en las cadenas (se empieza desde '\')
241	Se requiere una matriz - corchete cuadrado '[' no pertenece a la matriz, o como parámetro-matriz ofrecen no matriz
242	No puede ser inicializado mediante la sucesión inicializadora
243	Importación no definida
244	Error del optimizador en el árbol sintáctico
245	Han sido declaradas demasiadas estructuras (simplifique el programa)
246	Conversión del parámetro no está permitida
247	Uso incorrecto del operador delete
248	No se puede declarar un puntero a una referencia
249	No se puede declarar una referencia a una referencia
250	No se puede declarar un puntero a un puntero
251	Declaración de estructura en la lista de parámetros es inadmisible
252	Operación de conversión de tipos inadmisible
253	Un puntero puede ser declarado sólo para una clase o estructura
256	Identificador no declarado

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
257	Error del optimizador del código ejecutable
258	Error de generación del código ejecutable
260	Expresión inadmisible para el operador switch
261	Pool de constantes de cadenas está sobrelleno, simplifique el programa
262	Imposible convertir a una enumeración
263	No se puede usar virtual para los datos (miembros de clase o estructura)
264	No se puede llamar al método de clase protegido
265	Función virtual sobrescrita devuelve otro tipo
266	No se puede heredar una clase de una estructura
267	No se puede heredar una estructura de una clase
268	Constructor no puede ser virtual (especificador virtual es inadmisible)
269	Estructura no puede tener métodos virtuales
270	Una función debe tener curpo
271	Sobrecarga de funciones de sistema (funciones del terminal) está prohibida
272	Especificador const está prohibido para las funciones que no son miembros de una clase o estructura
274	No se puede cambiar miembros de clase en el método constante
276	Sucesión inicializadora inapropiada
277	Falta valor por defecto para el parámetro (particularidad de declaración de parámetros por defecto)
278	Sobrescritura de parámetro por defecto (diferentes valores en declaración y implementación)
279	No se puede llamar a método no constante para un objeto constante
280	Para acceder a los miembros se necesita un objeto (un punto está puesto para no clase/estructura)
281	No se puede usar el nombre de una estructura ya declarada para declaración
284	Conversión no permitida (con la herencia cerrada)
285	Estructuras y arrays no pueden ser usados como variables input
286	Especificador const es inadmisible para un constructor/destructor
287	Expresión de cadena incorrecto para el tipo datetime
288	Propiedad desconocida (#property)
289	Valor incorrecto para la propiedad

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
290	Índice incorrecto para la propiedad #property
291	Parámetro de llamada ha sido omitido - < func(x,) >
293	Objeto tiene que ser pasado por referencia
294	Array tiene que ser pasado por referencia
295	Función ha sido declarada como importada
296	Función ha sido declarada como exportada
297	No se puede exportar una función importada
298	Función importada no puede tener este parámetro (no se puede pasar puntero, clase o estructura que contiene un array dinámico, puntero, clase, etc.)
299	Tiene que ser una clase
300	Sección #import no está cerrada
302	Falta de correspondencia de tipos
303	extern-variable ya está inicializada
304	No se ha encontrado ni una función exportada o punto estándar de entrada
305	Prohibida la llamada explícita del constructor
306	El método ha sido declarado como método constante
307	El método no ha sido declarado como método constante
308	Tamaño incorrecto del archivo de recurso
309	Nombre del recurso incorrecto
310	Error de apertura del archivo de recurso
311	Error de lectura del archivo de recurso
312	Tipo de recurso desconocido
313	Ruta incorrecta hacia el archivo del recurso
314	El nombre especificado del recurso ya se utiliza
315	Se esperaban los parámetros de la macro
316	Tras el nombre de la macro debe ir el espacio
317	Error en la descripción de los parámetros de la macro
318	Número de parámetros inválido a la hora de usar la macro
319	Número máximo de parámetros (16) para la macro ha sido superado
320	La macro es muy complicada, hay que hacerla más simple
321	Sólo una enumeración puede ser el parámetro EnumToString()

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
322	Nombre del recurso es demasiado largo
323	Formato de imagen insoportable (se admite sólo el formato BMP con la profundidad del color de 24 o 32 bits)
324	Prohibido declarar un array dentro del operador
325	La función se puede definir sólo al nivel global
326	Esta declaración es inaceptable para el área de visibilidad actual (zona de declaración)
327	La inicialización de las variables estáticas con valores locales no está permitida
328	La declaración de un array de objetos que no disponen del constructor por defecto no está permitida
329	La lista de inicialización está permitida únicamente para los constructores
330	Falta la definición de la función después de la lista de inicialización
331	La lista de inicialización está vacía
332	La inicialización de un array en el constructor está prohibida
333	En la lista de inicialización está prohibido inicializar los miembros de la clase base
334	Se esperaba la expresión del tipo entero
335	El volumen de memoria requerido para el array supera el valor máximo permitido
336	El volumen de memoria requerido para la estructura supera el valor máximo permitido
337	El volumen de memoria requerido para las variables declaradas a nivel global supera el valor máximo permitido
338	El volumen de memoria requerido para las variables locales supera el valor máximo permitido
339	Constructor no definido
340	Nombre inválido para el archivo del icono
341	No se ha podido abrir el archivo del icono según la ruta especificada
342	El archivo del icono es incorrecto y no corresponde al formato ICO
343	La reinicialización del miembro en el constructor de la clase/estructura utilizando la lista de inicialización
344	La inicialización de los miembros estáticos en la lista de inicialización del constructor no está permitida
345	La inicialización del miembro no estático de la estructura/clase está prohibida a nivel global

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
346	El nombre del método de la clase/estructura coincide con el nombre del miembro declarado antes
347	El nombre del miembro de la clase/estructura coincide con el nombre del método declarado antes
348	Una función virtual no puede ser declarada como static
349	El modificador const no está permitido para una función estática
350	Un constructor o destructor no pueden ser estáticos
351	No se puede acceder a un miembro/método no estático de la clase o estructura desde una función estática
352	Tras la palabra clave operator se espera una operación de sobrecarga (+,-,[],++,-- etc.)
353	No todas las operaciones se puede sobrecargar en MQL5
354	La definición no corresponde a la declaración
355	Número de parámetros para el operador incorrecto
356	No se ha encontrado ninguna función del manejador de evento
357	No se puede exportar los métodos
358	No se puede normalizar el puntero a un objeto constante con el puntero a un objeto no constante
359	De momento las plantillas de las clases no se soportan
360	Sobrecarga de las plantillas de las funciones de momento no se soporta
361	Imposible aplicar la plantilla de la función
362	Parámetro ambiguo en la plantilla de la función (se encajan varios tipos de parámetros)
363	Imposible determinar con qué tipo de parámetro normalizar el argumento de la plantilla de la función
364	Número incorrecto de parámetros en la plantilla de la función
365	Plantilla de funciones no puede ser virtual
366	Plantillas de funciones no pueden ser exportadas
367	No se puede importar las plantillas de funciones
368	Estructuras que contienen objetos no están permitidas
369	Los arrays de cadenas y las estructuras que contienen los objetos no están permitidos
370	El miembro estático de la clase/estructura tiene que estar inicializado explícitamente

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
371	Limitación del compilador: la cadena no puede tener más de 65 535 caracteres
372	#ifdef/#endif incompatibles
373	El objeto de la clase no puede ser devuelto porque falta el constructor del copiado
374	No se puede usar los miembros no estáticos y/o métodos durante la inicialización de la variable estática
375	No se puede usar OnTesterInit() sin declarar el manejador OnTesterDeinit()
376	El nombre de la variable local coincide con el nombre de uno de los parámetros de la función
377	No se puede usar las macros __FUNCSIG__ y __FUNCTION__ fuera del cuerpo de la función
378	Tipo devuelto inválido. Por ejemplo, este error será mostrado para las funciones importadas desde DLL que devuelven una estructura o un puntero como resultado
379	Error al utilizar la plantilla
380	No se usa
381	Sintaxis no permitida al declarar la función virtual pura, solo se permite"=NULL" o "=0"
382	Solo las funciones virtuales pueden ser declaradas con el especificador puro ("=NULL" o "=0")
383	La clase abstracta no puede ser instanciada
384	Para la conversión dinámica con la ayuda del operador dynamic_cast, el tipo de designación debe ser un puntero al tipo de usuario
385	Se espera el tipo "puntero a una función"
386	No se da soporte a los punteros a métodos
387	Error, no ha sido posible determinar el tipo de puntero a una función
388	Conversión no disponible a causa de la herencia cerrada
389	La variable con el modificador const deberá ser inicializada al declararse
393	En la interfaz pueden ser declarados solo métodos con acceso público
394	Colocación no permitida de la interfaz en otra interfaz
395	La interfaz puede heredarse solo de otra interfaz
396	Esperando interfaz
397	Las interfaces dan soporte solo a la herencia pública
398	La interfaz no puede contener miembros
399	No es posible crear objetos de la interfaz directamente, solo a través por herencia

© 2000-2025, MetaQuotes Ltd.

Número	Descripción
400	No se puede usar el especificador en la declaración anticipada
401	No es posible la herencia de esta clase, dado que ha sido declarada con el especificador final
402	No es posible redefinir un método declarado con el especificador final
403	El especificador final se puede aplicar solo a las funciones virtuales
404	El método marcado con el especificador override, en realidad no se solapa con ninguna función de la clase básica
405	El especificador no está permitido en la definición de la función, solo en la declaración
406	No es posible convertir este tipo al indicado
407	Este tipo no se puede utilizar para la variable de recurso
408	Error en el archivo del proyecto
409	No puede ser utilizado como miembro union
410	Ambigüedad en la selección para el nombre dado, es necesario indicar explícitamente el contexto de uso
411	No es posible usar esta estructura de DLL
412	No es posible llamar una función marcada con el especificador delete
413	MQL4 no tiene soporte. Para compilar este programa, utilice el MetaEditor que se encuentra en la carpeta de instalación de su terminal MetaTrader 4

© 2000-2025, MetaQuotes Ltd.

Errores de tiempo de ejecución

GetLastError() es la función que devuelve el código del último error que se almacena en la variable predefinida _LastError. El valor de esta variable puede ser puesto a cero usando la función ResetLastError().

Constante	Valor	Descripción
ERR_SUCCESS	0	La operación se ha ejecutado con éxito
ERR_INTERNAL_ERROR	4001	Error interno inesperado
ERR_WRONG_INTERNAL_PARAMETER	4002	Parámetro erróneo durante la llamada built-in a la función del terminal de cliente
ERR_INVALID_PARAMETER	4003	Parámetro erróneo durante la llamada a la función de sistema
ERR_NOT_ENOUGH_MEMORY	4004	No hay memoria suficiente para ejecutar la función de sistema
ERR_STRUCT_WITHOBJECTS_ORCLASS	4005	Estructura contiene objetos de cadenas y/o de arrays dinámicos y/o estructuras con estos objetos y/o clases
ERR_INVALID_ARRAY	4006	Array del tipo inapropiado, tamaño inapropiado o objeto dañado del array dinámico
ERR_ARRAY_RESIZE_ERROR	4007	No hay memoria suficiente para la reubicación de un array, o un intento de cambio del tamaño de un array estático
ERR_STRING_RESIZE_ERROR	4008	No hay memoria suficiente para la reubicación de una cadena
ERR_NOTINITIALIZED_STRING	4009	Cadena no inicializada
ERR_INVALID_DATETIME	4010	Valor de fecha y/o hora incorrecto
ERR_ARRAY_BAD_SIZE	4011	El número total de los elementos del array no puede superar 2147483647
ERR_INVALID_POINTER	4012	Puntero erróneo
ERR_INVALID_POINTER_TYPE	4013	Tipo erróneo del puntero
ERR_FUNCTION_NOT_ALLOWED	4014	Función de sistema no está permitida para la llamada
ERR_RESOURCE_NAME_DUPLICATED	4015	Coincidencia del nombre del recurso dinámico y estático

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_RESOURCE_NOT_FOUND	4016	Recurso con este nombre no encontrado en EX5
ERR_RESOURCE_UNSUPPORTED_TYPE	4017	Tipo del recurso no soportado o el tamaño superior a 16 Mb
ERR_RESOURCE_NAME_IS_TOO_LONG	4018	Nombre del recurso supera 63 caracteres
ERR_MATH_OVERFLOW	4019	Desbordamiento (overflow) ocurrido al calcular la función
ERR_SLEEP_ERROR	4020	Fuera de la fecha de finalización de la simulación tras llamar a Sleep()
ERR_STOPPED	4022	La simulación ha sido interrumpida forzosamente desde el exterior. Por ejemplo, ha sido interrumpida por la optimización, o se ha cerrado la ventana de simulación visual, o se ha detenido el agente de simulación
ERR_INVALID_TYPE	4023	Tipo no válido
ERR_INVALID_HANDLE	4024	Manejador no válido
ERR_TOO_MANY_OBJECTS	4025	Pool de objetos lleno
Gráficos		
ERR_CHART_WRONG_ID	4101	Identificador erróneo del gráfico
ERR_CHART_NO_REPLY	4102	Gráfico no responde
ERR_CHART_NOT_FOUND	4103	Gráfico no encontrado
ERR_CHART_NO_EXPERT	4104	Gráfico no tiene un Asesor Experto que pueda procesar el evento
ERR_CHART_CANNOT_OPEN	4105	Error al abrir el gráfico
ERR_CHART_CANNOT_CHANGE	4106	Error al cambiar el símbolo y período del gráfico
ERR_CHART_WRONG_PARAMETER	4107	Valor erróneo del parámetro para la función de trabajo con los gráficos
ERR_CHART_CANNOT_CREATE_TIMER	4108	Error al crear el temporizador
ERR_CHART_WRONG_PROPERTY	4109	Identificador erróneo de la propiedad del gráfico

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_CHART_SCREENSHOT_FAILED	4110	Error al crear un screenshot
ERR_CHART_NAVIGATE_FAILED	4111	error de navegación por el gráfico
ERR_CHART_TEMPLATE_FAILED	4112	Error al aplicar una plantilla
ERR_CHART_WINDOW_NOT_FOUND	4113	Subventana que contiene el indicador especificado no encontrada
ERR_CHART_INDICATOR_CANNOT_ADD	4114	Error al insertar un indicador en el gráfico
ERR_CHART_INDICATOR_CANNOT_DEL	4115	Error al quitar un indicador desde el gráfico
ERR_CHART_INDICATOR_NOT_FOUND	4116	El indicador no ha sido encontrado en el gráfico especificado
Objetos gráficos		
ERR_OBJECT_ERROR	4201	Error al manejar un objeto gráfico
ERR_OBJECT_NOT_FOUND	4202	Objeto gráfico no encontrado
ERR_OBJECT_WRONG_PROPERTY	4203	Identificador erróneo de la propiedad del objeto gráfico
ERR_OBJECT_GETDATE_FAILED	4204	Imposible recibir fecha correspondiente al valor
ERR_OBJECT_GETVALUE_FAILED	4205	Imposible recibir valor correspondiente a la fecha
MarketInfo		
ERR_MARKET_UNKNOWN_SYMBOL	4301	Símbolo desconocido
ERR_MARKET_NOT_SELECTED	4302	Símbolo no está seleccionado en MarketWatch
ERR_MARKET_WRONG_PROPERTY	4303	Identificador erróneo de la propiedad del símbolo
ERR_MARKET_LASTTIME_UNKNOWN	4304	Hora del último tick no se conoce (no había ticks)
ERR_MARKET_SELECT_ERROR	4305	Error al agregar o eliminar el símbolo a/de MarketWatch
ERR_MARKET_SELECT_LIMIT	4306	Se ha superado el límite de símbolos seleccionados en MarketWatch
ERR_MARKET_SESSION_INDEX	4307	ID de sesión incorrecto al llamar a la función

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
		SymbolInfoSessionQuote/SymbolInfoSessionTrade
Acceso al historial		
ERR_HISTORY_NOT_FOUND	4401	Historial solicitado no encontrado
ERR_HISTORY_WRONG_PROPERTY	4402	Identificador erróneo de la propiedad del historial
ERR_HISTORY_TIMEOUT	4403	Se ha superado el límite de tiempo al solicitar la historia
ERR_HISTORY_BARS_LIMIT	4404	El número de barras solicitado está limitado por los ajustes del terminal
ERR_HISTORY_LOAD_ERRORS	4405	Errores múltiples al cargar la historia
ERR_HISTORY_SMALL_BUFFER	4407	La matriz receptora es demasiado pequeña para almacenar los datos solicitados
Global_Variables		
ERR_GLOBALVARIABLE_NOT_FOUND	4501	Variable global del terminal de cliente no encontrada
ERR_GLOBALVARIABLE_EXISTS	4502	Variable global del terminal de cliente con este nombre ya existe
ERR_GLOBALVARIABLE_NOT_MODIFIED	4503	Las variables globales no han sido modificadas
ERR_GLOBALVARIABLE_CANNOTREAD	4504	No ha sido posible abrir y leer el archivo con los valores de las variables globales
ERR_GLOBALVARIABLE_CANNOTWRITE	4505	No ha sido posible grabar el archivo con los valores de las variables globales
ERR_MAIL_SEND_FAILED	4510	Envío de carta fallido
ERR_PLAY_SOUND_FAILED	4511	Reproducción de sonido fallido
ERR_MQL5_WRONG_PROPERTY	4512	Identificador erróneo de la propiedad del programa
ERR_TERMINAL_WRONG_PROPERTY	4513	Identificador erróneo de la propiedad del terminal
ERR_FTP_SEND_FAILED	4514	Envío de archivo a través de ftp fallido

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_NOTIFICATION_SEND_FAILED	4515	No se ha podido enviar la notificación
ERR_NOTIFICATION_WRONG_PARAMETER	4516	Parámetro incorrecto para el envío de la notificación - en la función SendNotification() han pasado una línea vacía o NULL
ERR_NOTIFICATION_WRONG_SETTINGS	4517	Ajustes incorrectos de las notificaciones en el terminal (ID no especificada o permiso no concedido)
ERR_NOTIFICATION_TOO_FREQUENT	4518	Envío de notificaciones muy frecuente
ERR_FTP_NOSERVER	4519	No se ha indicado el servidor ftp en los ajustes
ERR_FTP_NOLOGIN	4520	No se ha indicado el login ftp en los ajustes
ERR_FTP_FILE_ERROR	4521	El archivo no existe
ERR_FTP_CONNECT_FAILED	4522	No ha sido posible conectarse al servidor ftp
ERR_FTP_CHANGEDIR	4523	En el servidor ftp no se ha encontrado el directorio para cargar el archivo
Buffers de indicadores personalizados		
ERR_BUFFERS_NO_MEMORY	4601	No hay memoria suficiente para la redistribución de buffers de indicadores
ERR_BUFFERS_WRONG_INDEX	4602	Índice erróneo de su búfer de indicadores
Propiedades de indicadores personalizados		
ERR_CUSTOM_WRONG_PROPERTY	4603	Identificador erróneo de la propiedad del indicador personalizado
Account		
ERR_ACCOUNT_WRONG_PROPERTY	4701	Identificador erróneo de la propiedad de la cuenta
ERR_TRADE_WRONG_PROPERTY	4751	Identificador erróneo de la propiedad de la actividad comercial

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_TRADE_DISABLED	4752	Prohibida la actividad comercial para el Asesor Experto
ERR_TRADE_POSITION_NOT_FOUND	4753	Posición no encontrada
ERR_TRADE_ORDER_NOT_FOUND	4754	Orden no encontrada
ERR_TRADE_DEAL_NOT_FOUND	4755	Transacción no encontrada
ERR_TRADE_SEND_FAILED	4756	Envío de solicitud comercial fallida
ERR_TRADE_CALC_FAILED	4758	Fallo al calcular el valor del beneficio o el margen
Indicadores		
ERR_INDICATOR_UNKNOWN_SYMBOL	4801	Símbolo desconocido
ERR_INDICATOR_CANNOT_CREATE	4802	No se puede crear indicador
ERR_INDICATOR_NO_MEMORY	4803	Memoria insuficiente para añadir el indicador
ERR_INDICATOR_CANNOT_APPLY	4804	Indicador no puede ser aplicado a otro indicador
ERR_INDICATOR_CANNOT_ADD	4805	Error al añadir indicador
ERR_INDICATOR_DATA_NOT_FOUND	4806	Datos solicitados no encontrados
ERR_INDICATOR_WRONG_HANDLE	4807	Handle del indicador es erróneo
ERR_INDICATOR_WRONG_PARAMETERS	4808	Número erróneo de parámetros al crear un indicador
ERR_INDICATOR_PARAMETERS_MISSING	4809	No hay parámetros cuando se crea un indicador
ERR_INDICATOR_CUSTOM_NAME	4810	El primer parámetro en la matriz tiene que ser el nombre del indicador personalizado
ERR_INDICATOR_PARAMETER_TYPE	4811	Tipo erróneo del parámetro en la matriz al crear un indicador
ERR_INDICATOR_WRONG_INDEX	4812	Índice del búfer de indicador que se solicita es erróneo
Profundidad de Mercado		
ERR_BOOKS_CANNOT_ADD	4901	No se puede añadir la profundidad de mercado
ERR_BOOKS_CANNOT_DELETE	4902	No se puede eliminar la profundidad de mercado

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_BOOKS_CANNOT_GET	4903	No se puede obtener los datos de la profundidad de mercado
ERR_BOOKS_CANNOT_SUBSCRIBE	4904	Error al suscribirse a la recepción de nuevos datos de la profundidad de mercado
Operaciones con archivos		
ERR_TOO_MANY_FILES	5001	No se puede abrir más de 64 archivos
ERR_WRONG_FILENAME	5002	Nombre del archivo no válido
ERR_TOO_LONG_FILENAME	5003	Nombre del archivo demasiado largo
ERR_CANNOT_OPEN_FILE	5004	Error al abrir el archivo
ERR_FILE_CACHEBUFFER_ERROR	5005	Memoria insuficiente para la caché de lectura
ERR_CANNOT_DELETE_FILE	5006	Error al eliminar el archivo
ERR_INVALID_FILEHANDLE	5007	Archivo con este manejador ya está cerrado, o no se abría en absoluto
ERR_WRONG_FILEHANDLE	5008	Manejador erróneo de archivo
ERR_FILE_NOTTOWRITE	5009	El archivo debe ser abierto para la escritura
ERR_FILE_NOTTOREAD	5010	El archivo debe ser abierto para la lectura
ERR_FILE_NOTBIN	5011	El archivo debe ser abierto como un archivo binario
ERR_FILE_NOTTXT	5012	El archivo debe ser abierto como un archivo de texto
ERR_FILE_NOTTXTORCSV	5013	El archivo debe ser abierto como un archivo de texto o CSV
ERR_FILE_NOTCSV	5014	El archivo debe ser abierto como un archivo CSV
ERR_FILE_READERROR	5015	Error de lectura de archivo
ERR_FILE_BINSTRINGSIZE	5016	Hay que especificar el tamaño de la cadena porque el archivo ha sido abierto como binario
ERR_INCOMPATIBLE_FILE	5017	Para los arrays de cadenas - un archivo de texto, para los demás -

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
		un archivo binario
ERR_FILE_IS_DIRECTORY	5018	No es un archivo, es un directorio
ERR_FILE_NOT_EXIST	5019	Archivo no existe
ERR_FILE_CANNOT_REWRITE	5020	No se puede reescribir el archivo
ERR_WRONG_DIRECTORYNAME	5021	Nombre erróneo del directorio
ERR_DIRECTORY_NOT_EXIST	5022	Directorio no existe
ERR_FILE_ISNOT_DIRECTORY	5023	Es un archivo, no es un directorio
ERR_CANNOT_DELETE_DIRECTORY	5024	No se puede eliminar el directorio
ERR_CANNOT_CLEAN_DIRECTORY	5025	No se puede limpiar el directorio (tal vez, uno o más archivos estén bloqueados y no se ha podido llevar a cabo la eliminación)
ERR_MQL_FILE_WRITEERROR	5026	No se ha podido escribir el recurso en el archivo
ERR_FILE_ENDOFFILE	5027	No se ha podido leer el siguiente fragmento de datos del archivo CSV (FileReadString, FileReadNumber, FileReadDatetime, FileReadBool), puesto que se ha alcanzado el final del archivo
Conversión de cadenas		
ERR_NO_STRING_DATE	5030	No hay fecha en la cadena
ERR_WRONG_STRING_DATE	5031	Fecha errónea en la cadena
ERR_WRONG_STRING_TIME	5032	Hora errónea en la cadena
ERR_STRING_TIME_ERROR	5033	Error de conversión de cadena a fecha
ERR_STRING_OUT_OF_MEMORY	5034	Memoria insuficiente para la cadena
ERR_STRING_SMALL_LEN	5035	Longitud de cadena es menos de la esperada
ERR_STRING_TOO_BIGNUMBER	5036	Número excesivamente grande, más que ULONG_MAX
ERR_WRONG_FORMATSTRING	5037	Cadena de formato errónea
ERR_TOO_MANY_FORMATTERS	5038	Hay más especificadores de formato que los parámetros

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_TOO_MANY_PARAMETERS	5039	Hay más Parámetros que los especificadores de formato
ERR_WRONG_STRING_PARAMETER	5040	Parámetro del tipo string dañado
ERR_STRINGPOS_OUTOFRANGE	5041	Posición fuera de los límites de la cadena
ERR_STRING_ZEROADDED	5042	Al final de la cadena se ha añadido 0, una operación inútil
ERR_STRING_UNKNOWNTYPE	5043	Tipo de datos desconocido durante la conversión a una cadena
ERR_WRONG_STRING_OBJECT	5044	Objeto de cadena dañado
Operaciones con matrices		
ERR_INCOMPATIBLE_ARRAYS	5050	Copiado de los arrays incompatibles. Un array de cadena puede ser copiado sólo en un array de cadena, un array numérico sólo en un array numérico
ERR_SMALL_ASSERIES_ARRAY	5051	El array que recibe está declarado como AS_SERIES, y no tiene el tamaño suficiente
ERR_SMALL_ARRAY	5052	Un array muy pequeño, posición de inicio está fuera de los límites del array
ERR_ZEROSIZE_ARRAY	5053	Un array de longitud cero
ERR_NUMBER_ARRAYS_ONLY	5054	Tiene que ser un array numérico
ERR_ONEDIM_ARRAYS_ONLY	5055	Tiene que ser un array unidimensional
ERR_SERIES_ARRAY	5056	No se puede usar serie temporal
ERR_DOUBLE_ARRAY_ONLY	5057	Tiene que ser un array del tipo double
ERR_FLOAT_ARRAY_ONLY	5058	Tiene que ser un array del tipo float
ERR_LONG_ARRAY_ONLY	5059	Tiene que ser un array del tipo long
ERR_INT_ARRAY_ONLY	5060	Tiene que ser un array del tipo int
ERR_SHORT_ARRAY_ONLY	5061	Tiene que ser un array del tipo short

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_CHAR_ARRAY_ONLY	5062	Tiene que ser un array del tipo char
ERR_STRING_ARRAY_ONLY	5063	Solo matrices del tipo string
Trabajo con OpenCL		
ERR_OPENCL_NOT_SUPPORTED	5100	Las funciones OpenCL no se soportan en este ordenador
ERR_OPENCL_INTERNAL	5101	Error interno al ejecutar OpenCL
ERR_OPENCL_INVALID_HANDLE	5102	Manejado OpenCL incorrecto
ERR_OPENCL_CONTEXT_CREATE	5103	Error al crear el contexto OpenCL
ERR_OPENCL_QUEUE_CREATE	5104	Error al crear la cola de ejecución en OpenCL
ERR_OPENCL_PROGRAM_CREATE	5105	Error al compilar el programa OpenCL
ERR_OPENCL_TOO_LONG_KERNEL_NAME	5106	Punto de entrada demasiado largo (kernel OpenCL)
ERR_OPENCL_KERNEL_CREATE	5107	Error al crear el kernel - punto de entrada de OpenCL
ERR_OPENCL_SET_KERNEL_PARAMETER	5108	Error al establecer los parámetros para el kernel OpenCL (punto de entrada en el programa OpenCL)
ERR_OPENCL_EXECUTE	5109	Error de ejecución del programa OpenCL
ERR_OPENCL_WRONG_BUFFER_SIZE	5110	Tamaño del búfer OpenCL incorrecto
ERR_OPENCL_WRONG_BUFFER_OFFSET	5111	Desplazamiento incorrecto en el búfer OpenCL
ERR_OPENCL_BUFFER_CREATE	5112	Error de creación del búfer OpenCL
ERR_OPENCL_TOO_MANY_OBJECTS	5113	Se ha superado el número de objetos OpenCL
ERR_OPENCL_SELECTDEVICE	5114	Error al elegir el dispositivo OpenCL
Trabajo con la base de datos		
ERR_DATABASE_INTERNAL	5120	Error interno en la base de datos
ERR_DATABASE_INVALID_HANDLE	5121	Puntero no válido de la base de datos

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_DATABASE_TOO_MANY_OBJECTS	5122	Se ha superado el número máximo permitido de objetos Database
ERR_DATABASE_CONNECT	5123	Error al conectarse a la base de datos
ERR_DATABASE_EXECUTE	5124	Error al ejecutar la solicitud
ERR_DATABASE_PREPARE	5125	Error al crear la solicitud
ERR_DATABASE_NO_MORE_DATA	5126	Ya no hay más datos para la lectura
ERR_DATABASE_STEP	5127	Error al pasar a la siguiente entrada de la solicitud
ERR_DATABASE_NOT_READY	5128	Los datos para leer los resultados de la solicitud aún no están preparados
ERR_DATABASE_BIND_PARAMETERS	5129	Error al realizar la autosustitución de los parámetros en la solicitud SQL
Trabajo con WebRequest		
ERR_WEBREQUEST_INVALID_ADDRESS	5200	URL no ha superado la prueba
ERR_WEBREQUEST_CONNECT_FAILED	5201	No se ha podido conectarse a la URL especificada
ERR_WEBREQUEST_TIMEOUT	5202	Superado el tiempo de espera de recepción de datos
ERR_WEBREQUEST_REQUEST_FAILED	5203	Error de ejecución de la solicitud HTTP
Trabajo con la red (sockets)		
ERR_NETSOCKET_INVALIDHANDLE	5270	A la función se ha transmitido un puntero incorrecto a del socket
ERR_NETSOCKET_TOO_MANY_OPENED	5271	Se han abierto demasiados sockets (128 como máximo)
ERR_NETSOCKET_CANNOT_CONNECT	5272	Error al conectar con el host remoto
ERR_NETSOCKET_IO_ERROR	5273	Error de envío/recepción de datos del socket
ERR_NETSOCKET_HANDSHAKE_FAILED	5274	Error al establecer una conexión protegida (TLS Handshake)
ERR_NETSOCKET_NO_CERTIFICATE	5275	No hay datos sobre el certificado con el que se protege la conexión

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
Símbolos personalizados		
ERR_NOT_CUSTOM_SYMBOL	5300	Debe indicarse un símbolo personalizado
ERR_CUSTOM_SYMBOL_WRONG_NAME	5301	Nombre del símbolo personalizado incorrecto. En el nombre dado al símbolo se usan solo letras latinas sin signos de puntuación, sin espacios en blanco y ni símbolos especiales (se permiten ".", "_", "&" y "#"). No se recomienda utilizar los símbolos <, >, :, ", /,\, |, ?, *.
ERR_CUSTOM_SYMBOL_NAME_LONG	5302	Nombre demasiado largo para el símbolo personalizado. La longitud del nombre no deberá superar los 32 caracteres contando el 0 final
ERR_CUSTOM_SYMBOL_PATH_LONG	5303	Ruta para el símbolo personalizado demasiado larga. La longitud de la ruta no deberá superar los 128 caracteres incluyendo "Custom\\", el nombre del símbolo, los separadores de grupos y el 0 final
ERR_CUSTOM_SYMBOL_EXIST	5304	No existe ningún símbolo personalizado con ese nombre
ERR_CUSTOM_SYMBOL_ERROR	5305	Error al crear, eliminar o modificar el símbolo personalizado
ERR_CUSTOM_SYMBOL_SELECTED	5306	Intento de eliminar un símbolo personalizado elegido en la observación de mercado (Market Watch)
ERR_CUSTOM_SYMBOL_PROPERTY_WRONG	5307	Propiedad de símbolo personalizado incorrecta
ERR_CUSTOM_SYMBOL_PARAMETER_ERROR	5308	Parámetro erróneo al establecer las propiedades del símbolo personalizado
ERR_CUSTOM_SYMBOL_PARAMETER_LONG	5309	Parámetro de cadena demasiado largo al establecer las propiedades del símbolo personalizado

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_CUSTOM_TICKS_WRONG_ORDER	5310	Matriz de ticks no organizada por tiempo
Calendario económico		
ERR_CALENDAR_MORE_DATA	5400	El tamaño de la matriz es insuficiente para obtener las descripciones de todos los valores
ERR_CALENDAR_TIMEOUT	5401	Se ha superado el límite de solicitud por tiempo
ERR_CALENDAR_NO_DATA	5402	El país no ha sido encontrado
Working with databases		
ERR_DATABASE_ERROR	5601	Generic error
ERR_DATABASE_LOGIC	5602	SQLite internal logic error
ERR_DATABASE_PERM	5603	Access denied
ERR_DATABASE_ABORT	5604	Callback routine requested abort
ERR_DATABASE_BUSY	5605	Database file locked
ERR_DATABASE_LOCKED	5606	Database table locked
ERR_DATABASE_NOMEM	5607	Insufficient memory for completing operation
ERR_DATABASE_READONLY	5608	Attempt to write to readonly database
ERR_DATABASE_INTERRUPT	5609	Operation terminated by sqlite3_interrupt()
ERR_DATABASE_IOERR	5610	Disk I/O error
ERR_DATABASE_CORRUPT	5611	Database disk image corrupted
ERR_DATABASE_NOTFOUND	5612	Unknown operation code in sqlite3_file_control()
ERR_DATABASE_FULL	5613	Insertion failed because database is full
ERR_DATABASE_CANTOPEN	5614	Unable to open the database file
ERR_DATABASE_PROTOCOL	5615	Database lock protocol error
ERR_DATABASE_EMPTY	5616	Internal use only
ERR_DATABASE_SCHEMA	5617	Database schema changed
ERR_DATABASE_TOOBIG	5618	String or BLOB exceeds size limit
ERR_DATABASE_CONSTRAINT	5619	Abort due to constraint violation

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_DATABASE_MISMATCH	5620	Data type mismatch
ERR_DATABASE_MISUSE	5621	Library used incorrectly
ERR_DATABASE_NOLFS	5622	Uses OS features not supported on host
ERR_DATABASE_AUTH	5623	Authorization denied
ERR_DATABASE_FORMAT	5624	Not used
ERR_DATABASE_RANGE	5625	Bind parameter error, incorrect index
ERR_DATABASE_NOTADB	5626	File opened that is not database file
Métodos de matrices y vectores		
ERR_MATRIX_INTERNAL	5700	Error interno del subsistema de ejecución de matrices/vectores
ERR_MATRIX_NOT_INITIALIZED	5701	La matriz/vector no ha sido inicializado
ERR_MATRIX_INCONSISTENT	5702	Tamaño de matriz/vector incoherente en la operación
ERR_MATRIX_INVALID_SIZE	5703	Tamaño de matriz/vector incorrecto
ERR_MATRIX_INVALID_TYPE	5704	Tipo de matriz/vector incorrecto
ERR_MATRIX_FUNC_NOT_ALLOWED	5705	La función no está disponible para esta matriz/vector
ERR_MATRIX_CONTAINS_NAN	5706	La matriz/vector contiene NaN (Nan/Inf)
Modelos ONNX		
ERR_ONNX_INTERNAL	5800	Error interno del estándar ONNX
ERR_ONNX_NOT_INITIALIZED	5801	Error de inicialización ONNX Runtime API
ERR_ONNX_NOT_SUPPORTED	5802	Propiedad o valor no compatible con el lenguaje MQL5
ERR_ONNX_RUN_FAILED	5803	Error de inicio ONNX runtime API
ERR_ONNX_INVALID_PARAMETERS_COUNT	5804	El número de parámetros transmitidos a OnnxRun es incorrecto
ERR_ONNX_INVALID_PARAMETER	5805	Valor de parámetro incorrecto

© 2000-2025, MetaQuotes Ltd.

Constante	Valor	Descripción
ERR_ONNX_INVALID_PARAMETER_TYPE	5806	Tipo de parámetro incorrecto
ERR_ONNX_INVALID_PARAMETER_SIZE	5807	Tamaño de parámetro incorrecto
ERR_ONNX_WRONG_DIMENSION	5808	La dimensionalidad del tensor no ha sido establecida o se ha indicado incorrectamente
ERR_USER_ERROR_FIRST	65536	A partir de este código se empiezan los errores definidos por el usuario

Véase también

Códigos de retorno del servidor comercial

© 2000-2025, MetaQuotes Ltd.

Constantes de entrada/salida

Constantes:

- Banderas de apertura de archivos
- Propiedades de archivos
- Posicionamiento dentro del archivo
- Uso de página de código
- MessageBox

© 2000-2025, MetaQuotes Ltd.

Banderas de apertura de archivos

Estos son los valores de banderas que determinan el modo de manejar un archivo. Las banderas están determinadas como sigue:

Identificador	Valor	Descripción
FILE_READ	1	Archivo se abre para la lectura. La bandera se usa cuando un archivo se abre, en la función (FileOpen()). Cuando abrimos un archivo, es obligatorio indicar la bandera FILE_WRITE y/o la bandera FILE_READ.
FILE_WRITE	2	Archivo se abre para la escritura. La bandera se usa cuando un archivo se abre, en la función (FileOpen()). Cuando abrimos un archivo, es obligatorio indicar la bandera FILE_WRITE y/o la bandera FILE_READ.
FILE_BIN	4	Modo binario de lectura-escritura (sin conversión de una cadena y en una cadena). La bandera se usa cuando un archivo se abre, en la función (FileOpen())
FILE_CSV	8	Archivo del tipo csv (todos sus elementos se convierten a las cadenas del tipo correspondiente, unicode o ansi, y se separan con un delimitador). La bandera se usa cuando un archivo se abre, en la función (FileOpen())
FILE_TXT	16	Archivo de texto simple (igual que el archivo csv pero sin tomar en cuenta los delimitadores). La bandera se usa cuando un archivo se abre, en la función (FileOpen())
FILE_ANSI	32	Cadenas del tipo ANSI (símbolos de un byte). La bandera se usa cuando un archivo se abre, en la función (FileOpen())
FILE_UNICODE	64	Cadenas del tipo UNICODE (símbolos de dos byte). La bandera se usa cuando un archivo se abre, en la función (FileOpen())
FILE_SHARE_READ	128	Acceso compartido para la lectura desde varios programas. La bandera se usa cuando se abre un archivo (FileOpen()), pero durante la apertura no sustituye la necesidad de indicar FILE_WRITE y/o la bandera FILE_READ
FILE_SHARE_WRITE	256	Acceso compartido para la escritura desde varios programas. La bandera se usa cuando se abre un archivo (FileOpen()), pero durante la apertura no sustituye la necesidad de indicar FILE_WRITE y/o la bandera FILE_READ
FILE_REWRITE	512	Posibilidad de reescribir un archivo usando las funciones FileCopy() y FileMove(). El archivo tiene que existir o abrirse para la escritura. En caso contrario el archivo no se abrirá.

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor	Descripción
FILE_COMMON	4096	Ubicación del archivo en la carpeta general de todos los terminales de cliente \Terminal\Common\Files. La bandera se usa en las funciones (FileOpen()), (FileCopy(), FileMove()), (FileIsExist())

Durante la apertura de un archivo se puede indicar una o más banderas. Esto se llama la combinación de banderas. Esta combinación se escribe mediante el símbolo del lógico OR (|), éste se coloca entre las banderas especificadas. Por ejemplo, para abrir un archivo en el formato CSV para la lectura y escritura al mismo tiempo, podemos indicar la combinación FILE_READ|FILE_WRITE|FILE_CSV.

Ejemplo:

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

Existen unas particularidades a la hora de indicar las banderas de lectura y escritura:

- Si indicamos FILE_READ, se intenta abrir un archivo que ya existe. Si este archivo no existe, no se puede abrirlo y el nuevo no se crea.
- Si ponemos FILE_READ|FILE_WRITE, se crea un archivo nuevo, si el archivo con el nombre especificado no existe.
- FILE_WRITE - el archivo vuelve a crearse teniendo el tamaño cero.

Cuando abrimos un archivo, es obligatorio indicar la bandera FILE_WRITE y/o la bandera FILE_READ.

Las banderas que determinan el tipo de lectura de un archivo abierto poseen la prioridad. La mayor prioridad tiene la bandera FILE_CSV, luego FILE_BIN, y luego FILE_TXT que tiene la menor prioridad. De esta manera, si están indicadas varias banderas (FILE_TXT|FILE_CSV o FILE_TXT|FILE_BIN o FILE_BIN|FILE_CSV), se usará la bandera con la mayor prioridad.

Las bandera que determinan el tipo de codificación también tienen la prioridad. La bandera FILE_UNICODE es de mayor prioridad que la bandera FILE_ANSI. Por tanto, al indicar la combinación FILE_UNICODE|FILE_ANSI, se usa la bandera FILE_UNICODE.

Si no se indica FILE_UNICODE, ni tampoco FILE_ANSI, se sobreentiende FILE_UNICODE. Si no se indica FILE_CSV, ni FILE_BIN, ni FILE_TXT, entonces se sobreentiende FILE_CSV.

Si un archivo está abierto para la lectura como un archivo de texto (FILE_TXT o FILE_CSV), y además, al principio de este archivo se encuentra 0xff,0xfe de dos bytes, la bandera de codificación será FILE_UNICODE, incluso si se especifica la bandera FILE_ANSI.

Véase también

Operaciones con archivos

© 2000-2025, MetaQuotes Ltd.

Propiedades de archivos

Para obtener las propiedades de archivos se utiliza la función FileGetInteger(). Durante la llamada se le pasa el identificador de la propiedad requerida desde la enumeración ENUM_FILE_PROPERTY_INTEGER

ENUM_FILE_PROPERTY_INTEGER

Identificador	Descripción del identificador
FILE_EXISTS	Comprobación de existencia
FILE_CREATE_DATE	Fecha de creación
FILE_MODIFY_DATE	Fecha de última modificación
FILE_ACCESS_DATE	Fecha del último acceso al archivo
FILE_SIZE	Tamaño del archivo en bytes
FILE_POSITION	Posición del puntero en el archivo
FILE_END	Obtención del signo del final del archivo
FILE_LINE_END	Obtención del signo del final de la línea
FILE_IS_COMMON	El archivo está abierto en la carpeta común de todos los terminales de cliente (ver FILE_COMMON)
FILE_IS_TEXT	El archivo está abierto como un archivo de texto (ver FILE_TXT)
FILE_IS_BINARY	El archivo está abierto como un archivo binario (ver FILE_BIN)
FILE_IS_CSV	El archivo está abierto como un archivo CSV (ver FILE_CSV)
FILE_IS_ANSI	El archivo está abierto como un archivo ANSI (ver FILE_ANSI)
FILE_IS_READABLE	El archivo está abierto con posibilidades de lectura (ver FILE_READ)
FILE_IS_WRITABLE	El archivo está abierto con posibilidades de escritura (ver FILE_WRITE)

La función FileGetInteger() tiene dos diferentes opciones de llamada. En la primera, para obtener las propiedades del archivo, se indica su manejador obtenido durante la apertura del archivo por medio de la función FileOpen(). Esta variante permite obtener todas las propiedades del archivo.

La segunda opción de la función FileGetInteger() devuelve los valores de las propiedades del archivo por su nombre. Esta opción permite obtener sólo las siguientes propiedades comunes:

- FILE_EXISTS - existencia del archivo con el nombre especificado;
- FILE_CREATE_DATE - fecha de creación del archivo con el nombre especificado;
- FILE_MODIFY_DATE - fecha de modificación del archivo con el nombre especificado;
- FILE_ACCESS_DATE - fecha del último acceso al archivo con el nombre especificado;
- FILE_SIZE - tamaño del archivo con el nombre especificado.

Si intenta obtener otras propiedades aparte de las arriba mencionadas, la segunda variante de la llamada a la función FileGetInteger() devolverá un error.

© 2000-2025, MetaQuotes Ltd.

Posicionamiento dentro del archivo

La mayor parte de las funciones de archivos está vinculada con las operaciones de lectura/escritura de información. Usando la función FileSeek() podemos especificar la posición de un puntero de archivos sobre una posición dentro del archivo a partir de la cual va a realizarse la siguiente operación de lectura o escritura. La enumeración ENUM_FILE_POSITION contiene las posiciones válidas de un puntero respecto al que podemos especificar el desplazamiento en bytes para la operación siguiente.

ENUM_FILE_POSITION

Identificador	Descripción
SEEK_SET	Principio del archivo
SEEK_CUR	Posición actual de un puntero de archivos
SEEK_END	Final del archivo

Véase también

FileIsEnding, FileIsLineEnding

© 2000-2025, MetaQuotes Ltd.

Uso de la página de código en las operaciones de conversión de cadenas

En el lenguaje MQL5, durante las operaciones de conversión de variables de cadenas a los arrays del tipo char y viceversa, se utiliza la codificación que por defecto corresponde a la actual codificación ANSI del sistema operativo Windows (CP_ACP). Si hace falta especificar otro tipo de codificación, se puede definirlo usando un parámetro adicional para las funciones CharArrayToString(), StringToCharArray() y FileOpen().

En la tabla de abajo vienen las constantes built-in para algunas de las páginas de códigos más usadas. Las páginas de códigos que no vienen en esta tabla pueden ser definidas con el código correspondiente a esta página.

Constantes built-in de las páginas de códigos

Constante	Valor	Descripción
CP_ACP	0	Página de código actual de codificación ANSI en el sistema operativo Windows
CP_OEMCP	1	Página de código actual OEM.
CP_MACCP	2	Página de código actual Macintosh. **Nota:** Este valor suelen usar en los códigos de programas creados anteriormente, y actualmente no hay necesidad de usarlo porque los modernos ordenadores Macintosh utilizan la codificación Unicode.
CP_THREAD_ACP	3	Codificación Windows ANSI para el hilo de ejecución corriente.
CP_SYMBOL	42	Página de código Symbol
CP_UTF7	65000	Página de código UTF-7.
CP_UTF8	65001	Página de código UTF-8.

Véase también

Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

Constantes de la ventana de diálogo MessageBox

Códigos de retorno de la función MessageBox(). Si la ventana de mensaje dispone del botón Cancelar (Cancel), la función devuelve el valor IDCANCEL al apretar la tecla ESC o al pulsar el botón Cancelar (Cancel). Si la ventana de mensaje no dispone del botón Cancelar (Cancel), pulsando ESC no provoca efecto alguno.

Constante	Valor	Descripción
IDOK	1	El botón "OK" está seleccionado
IDCANCEL	2	El botón "Cancelar" (Cancel) está seleccionado
IDABORT	3	El botón "Interrumpir" (Abort) está seleccionado
IDRETRY	4	El botón "Reintentar" (Retry) está seleccionado
IDIGNORE	5	El botón "Ignorar" (Ignore) está seleccionado
IDYES	6	El botón "Sí" (Yes) está seleccionado
IDNO	7	El botón "No" (No) está seleccionado
IDTRYAGAIN	10	El botón "Repetir" (Try Again) está seleccionado
IDCONTINUE	11	El botón "Continuar" (Continue) está seleccionado

Las banderas principales de la función MessageBox() definen el contenido y comportamiento de la ventana de diálogo. Este valor puede ser una combinación de los siguientes grupos de banderas:

Constante	Valor	Descripción
MB_OK	0x00000000	La ventana de diálogo contiene un botón: OK. Por defecto
MB_OKCANCEL	0x00000001	La ventana de diálogo contiene dos botones: OK y Cancel
MB_ABORTRETRYIGNORE	0x00000002	La ventana de diálogo contiene tres botones: Abort, Retry y Ignore
MB_YESNOCANCEL	0x00000003	La ventana de diálogo contiene tres botones: Yes, No y Cancel
MB_YESNO	0x00000004	La ventana de diálogo contiene dos botones: Yes y No
MB_RETRYCANCEL	0x00000005	La ventana de diálogo contiene dos botones: Retry y Cancel
MB_CANCELTRYCONTINUE	0x00000006	La ventana de diálogo contiene tres botones: Cancel, Try Again, Continue

© 2000-2025, MetaQuotes Ltd.

Para mostrar un ícono en la ventana de diálogo es necesario especificar las banderas adicionales:

Constante	Valor	Descripción
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	0x00000010	El ícono del signo STOP
MB_ICONQUESTION	0x00000020	El ícono del signo interrogación
MB_ICONEXCLAMATION, MB_ICONWARNING	0x00000030	El ícono del signo de admiración
MB_ICONINFORMATION, MB_ICONASTERISK	0x00000040	El ícono del signo **i** dentro del círculo

Los botones predefinidos se definen mediante las siguientes banderas:

Constante	Valor	Descripción
MB_DEFBUTTON1	0x00000000	El primer botón MB_DEFBUTTON1 - el botón está seleccionado por defecto, si MB_DEFBUTTON2, MB_DEFBUTTON3, o MB_DEFBUTTON4 no están especificados
MB_DEFBUTTON2	0x00000100	El segundo botón - botón por defecto
MB_DEFBUTTON3	0x00000200	El tercer botón - botón por defecto
MB_DEFBUTTON4	0x00000300	El cuarto botón - botón por defecto

© 2000-2025, MetaQuotes Ltd.

Programas MQL5

Para que el programa mql5 pueda funcionar, tiene que estar compilado (botón "Compilar" o tecla F7). La compilación debe pasar sin errores(se admiten advertencias que hace falta analizar). Durante este proceso hay que crear un archivo ejecutable con el mismo nombre y la extensión EX5 en el directorio correspondiente, *terminal_dir\MQL5\Experts*, *terminal_dir\MQL5\indicators* o *terminal_dir\MQL5\scripts*. Precisamente este archivo puede ser ejecutado.

Las particularidades de funcionamiento de los programas mql5 se describen en los siguientes apartados:

- Ejecución de programas – orden de llamada a las funciones-manejadores de eventos predefinidas;
- Prueba de estrategias de trading – particularidades de funcionamiento de programas en el Probador de Estrategias;
- Eventos de terminal de cliente – descripción de eventos que pueden ser procesados en los programas;
- Llamada a las funciones importadas – orden de descripción, parámetros admisibles, orden de búsqueda y acuerdos de enlaces de las funciones importadas;
- Errores de ejecución – obtención de información sobre los errores de ejecución y errores críticos.

Los Asesores Expertos, indicadores personalizados y scripts se adjuntan a uno de los gráficos abiertos arrastrándolos con el ratón desde la ventana "Navegador" del terminal de cliente hasta el gráfico correspondiente (tecnología Drag'n'Drop). Los programas mql5 pueden trabajar sólo con el terminal de cliente que esté en funcionamiento.

Para que el Asesor Experto deje de trabajar, hay que eliminarlo del gráfico seleccionando "Asesores Expertos - Eliminar" del menú contextual de gráfico. Además, el estado del botón "Activar/desactivar Asesores Expertos" también influye en el funcionamiento de Asesor.

Para que el indicador de usuario deje de trabajar, hay que eliminarlo del gráfico.

Los indicadores personalizados y Asesores Expertos están operativos hasta que no sean eliminados explícitamente del gráfico; la información sobre los Asesores Expertos e indicadores de usuario adjuntos se guarda entre los inicios del terminal de cliente.

Los scripts se ejecutan sólo una vez y se eliminan automáticamente al terminar su trabajo o después de que el estado del gráfico corriente haya sido cerrado o cambiado, o después del fin de funcionamiento del terminal de cliente. Durante el reinicio del terminal de cliente los scripts no se inician porque la información sobre ellos no se guarda.

Como máximo un Asesor Experto, un script y el número ilimitado de indicadores pueden estar operativos en un gráfico.

Para su trabajo, los servicios no requieren vinculación alguna al gráfico, y han sido pensados para ejecutar funciones auxiliares. Por ejemplo, en el servicio se puede crear un símbolo personalizado, abrir el gráfico del símbolo creado, obtener después los datos para él en un ciclo infinito con la ayuda de funciones de red y actualizar ininterrumpidamente.

© 2000-2025, MetaQuotes Ltd.

Ejecución de programas

Cada script, servicio y experto trabaja en su propio flujo de ejecución independiente. Todos los indicadores que se calculan para un símbolo trabajan en un flujo de ejecución, incluso si han sido iniciados en diferentes gráficos. De esta manera, todos los indicadores de un símbolo comparten entre ellos los recursos de un flujo de ejecución.

Todas las demás acciones relacionadas con este símbolo (procesamiento de los ticks y sincronización del historial) también se ejecutan sucesivamente en el mismo flujo junto con los indicadores. Eso quiere decir que si en un indicador se realiza una acción infinita, todos los demás eventos para este símbolo nunca se ejecutan.

Cuando se arranca un Asesor Experto, es necesario asegurar que disponga de un entorno de trading puesto al día, que pueda acceder al historial para este símbolo y período, así como realizar la sincronización entre el terminal y el servidor. Para estos procedimientos el terminal concede al EA un retraso de arranque de no más de 5 segundos, una vez expirados los cuales, el EA será iniciado con los datos que se ha podido preparar. Por eso en caso de no haber conexión con el servidor, esto puede provocar el retraso de arranque del EA.

La tabla de abajo contiene un breve sumario para los programas escritos en MQL5:

Programa	Ejecución	Nota
Servicio	En un flujo propio; el número de servicios será igual que el número de flujos de ejecución para los mismos	Un ciclo infinito no puede parar el trabajo de otros programas
Script	En un flujo separado el número de flujos para los scripts coincide con el número de los scripts	Un script de ciclo cerrado no puede alterar el funcionamiento de otros programas
Asesor Experto	En un flujo separado el número de flujos para los EAs coincide con el número de los EAs	Un Asesor Experto de ciclo cerrado no puede alterar el funcionamiento de otros programas
Indicador	Un flujo de ejecución para todos los indicadores en un símbolo. El número de flujos de ejecución para los indicadores coincide con el número de los símbolos con estos indicadores	Un ciclo infinito en un indicador parará el trabajo de todos los demás indicadores en este símbolo

Inmediatamente después de que un programa haya sido adjuntado a un gráfico, este programa se carga en la memoria del terminal de cliente y se realiza la inicialización de variables globales. Si alguna variable global del tipo de clase dispone del constructor, éste va a ser invocado durante el proceso de inicialización de variables globales.

Después de todo eso el programa se encuentra en el modo de espera de un evento del terminal de cliente. Cada programa mql5 debe tener por lo menos una función-manejador de eventos, en caso

© 2000-2025, MetaQuotes Ltd.

contrario, el programa cargado no será ejecutado. Las funciones-manejadores de eventos tienen los nombres predefinidos, conjuntos de parámetros y los tipos de retorno predefinidos.

Tipo	Nombre de la función	Parámetros	Aplicación	Comentario
int	OnInit	no hay	expertos e indicadores	Manejador de evento Init. Se admite el tipo de valor devuelto void.
void	OnDeinit	const int reason	expertos e indicadores	Manejador de evento Deinit.
void	OnStart	no hay	scripts y servicios	Manejador de evento Start.
int	OnCalculate	const int rates_total, const int prev_calculated, const datetime &Time[], const double &Open[], const double &High[], const double &Low[], const double &Close[], const long &TickVolume[], const long &Volume[], const int &Spread[]	indicadores	Manejador de evento Calculate para todos los datos de precio.
int	OnCalculate	const int rates_total, const int prev_calculated, const int begin, const double &price[]	indicadores	Manejador de evento Calculate en un array de datos. No se permite el uso de dos manejadores Calculate en un indicador a la vez. En este caso sólo un manejador de evento Calculate va a funcionar en un array de datos.
void	OnTick	no hay	expertos	Manejador de evento NewTick. Mientras se realiza el procesamiento del evento de entrada de un nuevo tick, otros eventos de este tipo no pueden entran.

© 2000-2025, MetaQuotes Ltd.

Tipo	Nombre de la función	Parámetros	Aplicación	Comentario
void	OnTimer	no hay	expertos e indicadores	Manejador de evento Timer.
void	OnTrade	no hay	expertos	Manejador de evento Trade.
double	OnTester	no hay		Manejador de evento Tester
void	OnChartEvent	const int id, const long &lparam, const double &dparam, const string &sparam	expertos e indicadores	Manejador de evento ChartEvent.
void	OnBookEvent	const string &symbol_name	expertos e indicadores	Manejador de evento BookEvent.

El terminal de cliente envía los eventos que surgen a los correspondientes gráficos abiertos. Además, los eventos también pueden ser generados por los gráficos (eventos del gráfico) o por los programas mql5 (eventos de usuario). Usted puede activar y desactivar la generación de los eventos de creación y eliminación de los objetos gráficos ajustando las propiedades del gráfico CHART_EVENT_OBJECT_CREATE y CHART_EVENT_OBJECT_DELETE. Cada programa mql5 y cada gráfico tiene su propia cola de eventos en la que se ponen todos los eventos recién llegados.

El programa recibe los eventos sólo del gráfico en el que está iniciado. Todos los eventos se procesan uno tras otro, en orden de su llegada. Si en la cola ya hay un evento NewTick, o este evento se encuentra en el proceso de tramitación, entonces el nuevo evento NewTick no se coloca en la cola del programa mql5. Del mismo modo, si la cola del programa mql5 ya contiene un evento ChartEvent, o este evento se está procesando, el nuevo evento de este tipo no se coloca en la cola. El procesamiento del evento del temporizador se realiza según el mismo esquema. Es decir, si el evento Timer se encuentra en la cola o se está procesando, entonces el nuevo evento del temporizador no se pone en la cola.

Las colas de eventos tienen un tamaño limitado pero es suficiente, por eso la situación del desbordamiento de la cola es poco probable para un programa escrito de forma correcta. En caso del desbordamiento de la cola los nuevos eventos se descartan sin ponerse a la cola.

No se recomienda en ningún caso usar ciclos infinitos para el procesamiento de eventos. La única excepción a esta regla pueden ser solo los scripts y servicios que procesan un solo evento Start.

Las bibliotecas no manejan ningunos eventos.

© 2000-2025, MetaQuotes Ltd.

Prohibición para el uso de funciones en los indicadores y EAs

Los indicadores, scripts y EAs son programas ejecutables en MQL5 y están destinados para diferentes tipos de tareas. Por esta razón existe una restricción de uso de ciertas funciones dependiendo del tipo del programa. En los indicadores están prohibidas las funciones siguientes:

- OrderCalcMargin();
- OrderCalcProfit();
- OrderCheck();
- OrderSend();
- SendFTP();
- Sleep();
- ExpertRemove();
- MessageBox().

A su vez, en los EAs y scripts están prohibidas todas las funciones destinadas para los indicadores:

- SetIndexBuffer();
- IndicatorSetDouble();
- IndicatorSetInteger();
- IndicatorSetString();
- PlotIndexSetDouble();
- PlotIndexSetInteger();
- PlotIndexSetString();
- PlotIndexGetInteger.

La biblioteca no es un programa independiente y se ejecuta en el contexto del programa MQL5 que la ha llamado: un script, indicador o EA. En consecuencia, las restricciones especificadas conciernen a la biblioteca llamada.

Prohibición del uso de funciones en los servicios

Los servicios no aceptan ningún evento, ya que no están vinculados a un gráfico. En los servicios están prohibidas las siguientes funciones:

ExpertRemove();

EventSetMillisecondTimer();

EventSetTimer();

EventKillTimer();

SetIndexBuffer();

IndicatorSetDouble();

© 2000-2025, MetaQuotes Ltd.

IndicatorSetInteger();

IndicatorSetString();

PlotIndexSetDouble();

PlotIndexSetInteger();

PlotIndexSetString();

PlotIndexGetInteger();

Carga y descarga de indicadores

Los indicadores se cargan en siguientes ocasiones:

- indicador se adjunta al gráfico;
- Inicio del terminal (si el indicador estaba adjuntado al gráfico antes del cierre anterior de terminal);
- carga de plantilla (si el indicador adjunto al gráfico está especificado en la plantilla);
- cambio de perfil (si el indicador está adjuntado a uno de los gráficos del perfil);
- cambio del símbolo o/y período del gráfico al que está adjuntado el indicador;
- cambio de la cuenta a la que estaba conectado el terminal;
- después de recompilación del indicador finalizada con éxito, si este indicador ha sido adjuntado al gráfico.
- cambio de parámetros de entrada del indicador.

Los indicadores se descargan en siguientes ocasiones:

- si el indicador se desjunta del gráfico;
- cierre del terminal (si el indicador estaba adjuntado al gráfico);
- carga de plantilla, si el indicador está adjuntado al gráfico;
- cierre del gráfico al que ha sido adjuntado el indicador;
- cambio de perfil, si el indicador está adjuntado a uno de los gráficos del perfil que se cambia;
- cambio del símbolo o/y período del gráfico al que está adjuntado el indicador;
- cambio de la cuenta a la que estaba conectado el terminal;
- cambio de parámetros de entrada del indicador.

Carga y descarga de Asesores Expertos

Los Asesores Expertos se cargan en siguientes ocasiones:

- Asesor Experto se adjunta al gráfico;
- Inicio del terminal (si el Asesor Experto ha sido adjuntado al gráfico antes del cierre anterior de terminal);
- carga de plantilla (si el Asesor Experto adjunto al gráfico está especificado en la plantilla);

© 2000-2025, MetaQuotes Ltd.

- después de recompilación del Asesor Experto finalizada con éxito, si este Asesor Experto ha sido adjuntado al gráfico;
- cambio de perfil (si el Asesor Experto está adjuntado a uno de los gráficos del perfil);
- conexión a una cuenta, incluso si el número de la cuenta es el mismo (si el Asesor Experto ha sido adjuntado al gráfico antes de la autorización del terminal en el servidor).

La descarga del Asesor Experto adjunto al gráfico se realiza en las siguientes ocasiones:

- si el Asesor Experto se desjunta del gráfico;
- cuando el Asesor Experto se adjunta al gráfico - si ya había otro Asesor Experto en el mismo gráfico, éste se descarga ;
- el cierre del terminal (si el Asesor Experto estaba adjuntado al gráfico);
- carga de plantilla, si el Asesor Experto está adjuntado al gráfico;
- cierre del gráfico al que ha sido adjuntado el Asesor Experto;
- cambio de perfil, si el Asesor Experto está adjuntado a uno de los gráficos del perfil que se cambia;
- cambio de la cuenta a la que estaba conectado el terminal (si el Asesor Experto ha sido adjuntado al gráfico antes de la autorización del terminal en el servidor);
- calling the ExpertRemove() function.

Si el símbolo o período del gráfico al que el Experto está adjuntado, se cambia, entonces la carga y descarga del Asesor Experto no se realiza. En este caso, sucesivamente se invocan los manejadores OnDeinit() en el símbolo/período antiguo y OnInit() en el símbolo/período nuevo (si hay), los valores de variables globales y variables estáticas no se ponen a cero. Todos los eventos que han llegado para el Asesor Experto antes de completarse la inicialización (función OnInit()) se saltan.

Carga y descarga de scripts

Los scripts se cargan una vez adjuntados al gráfico, y se descargan inmediatamente al terminar su trabajo. Las funciones OnInit() y OnDeinit() no se invocan para los scripts.

Cuando el programa se descarga (se elimina del gráfico), ocurre deinicialización de variables globales y eliminación de la cola de mensajes. En este caso la deinicialización significa desasignación de variables del tipo string, liberación de objetos de arrays dinámicos y llamada a los destructores, en caso de que hayan.

Carga y descarga de servicios

Los servicios se cargan justo después del inicio del terminal, si en el momento en que se ha detenido el terminal, estos estaban iniciados. Los servicios se descargan inmediatamente después de su funcionamiento.

Los servicios tienen un único manejador OnStart(), en él usted podrá organizar un ciclo constante de obtención y procesamiento de datos, por ejemplo, para crear y actualizar los símbolos personalizados con la ayuda de las funciones de red.

A diferencia de los asesores, indicadores y scripts, los servicios no están vinculados a un gráfico concreto, por eso, para iniciar el servicio se ha pensado un mecanismo aparte. La creación de un

© 2000-2025, MetaQuotes Ltd.

nuevo ejemplar del servicio se realiza desde el Navegador, con la ayuda del comando "Añadir servicio". Para iniciar, detener y eliminar un ejemplar del servicio, utilice el menú del mismo. Para gestionar todos los ejemplares, utilice el menú del propio servicio.

Para mejor entendimiento de funcionamiento de los Asesores Expertos se recomienda hacer la compilación del código del Asesor Experto propuesto en el ejemplo y realizar las acciones de carga/descarga de Expertos, cambio de plantilla, símbolo, período, etc.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   TestExpert.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

class CTestClass
  {
public:
   CTestClass() { Print("CTestClass constructor"); }
   ~CTestClass() { Print("CTestClass destructor"); }
  };
CTestClass global;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Initialisation");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print("Deinitialisation with reason ",reason);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
```

Los scripts se cargan una vez adjuntados al gráfico, y se descargan al terminar su trabajo.

Véase también

© 2000-2025, MetaQuotes Ltd.

Eventos del terminal de cliente, Funciones de procesamiento de eventos

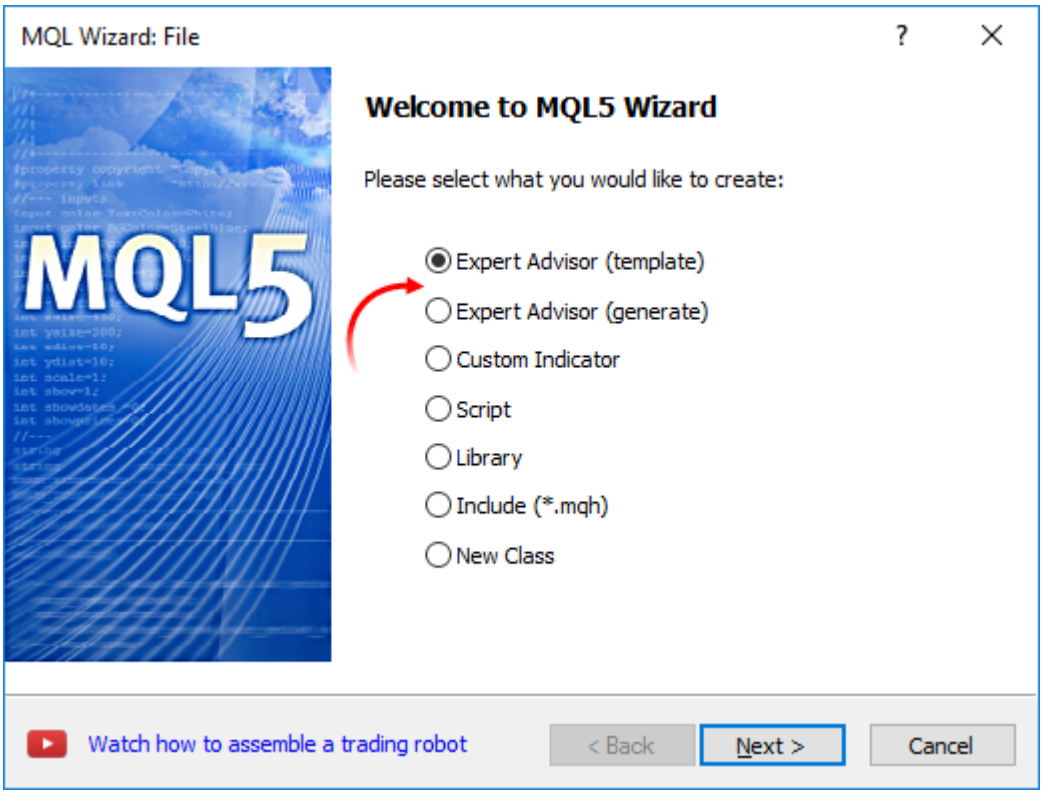
© 2000-2025, MetaQuotes Ltd.

Permiso para tradear

Automatización del trading

En el lenguaje MQL5 hay un grupo especial de funciones comerciales que permiten crear los sistemas comerciales automatizados. Los programas para el trading automático sin la intervención del hombre se llaman los Asesores Expertos (Expert Advisor) o robots comerciales. Para crear un EA, inicie en el editor MetaEditor el asistente MQL5 Wizard para el desarrollo de los EAs y seleccione una de dos opciones:

- Expert Advisor (template) – permite crear una plantilla que ya contiene las funciones de procesamiento de eventos hechas y que hace falta completar con todas las capacidades funcionales necesarias mediante la programación personal.
- Expert Advisor (generate) – permite desarrollar un robot comercial completamente funcional y listo para el trabajo simplemente seleccionando los módulos necesarios: módulo de formación de señales comerciales, módulo de gestión del capital y el módulo de arrastre del stop de protección para las posiciones abiertas (Trailing Stop).



Las funciones comerciales trabajan sólo en los EAs y los scripts, el trading está prohibido para los indicadores.

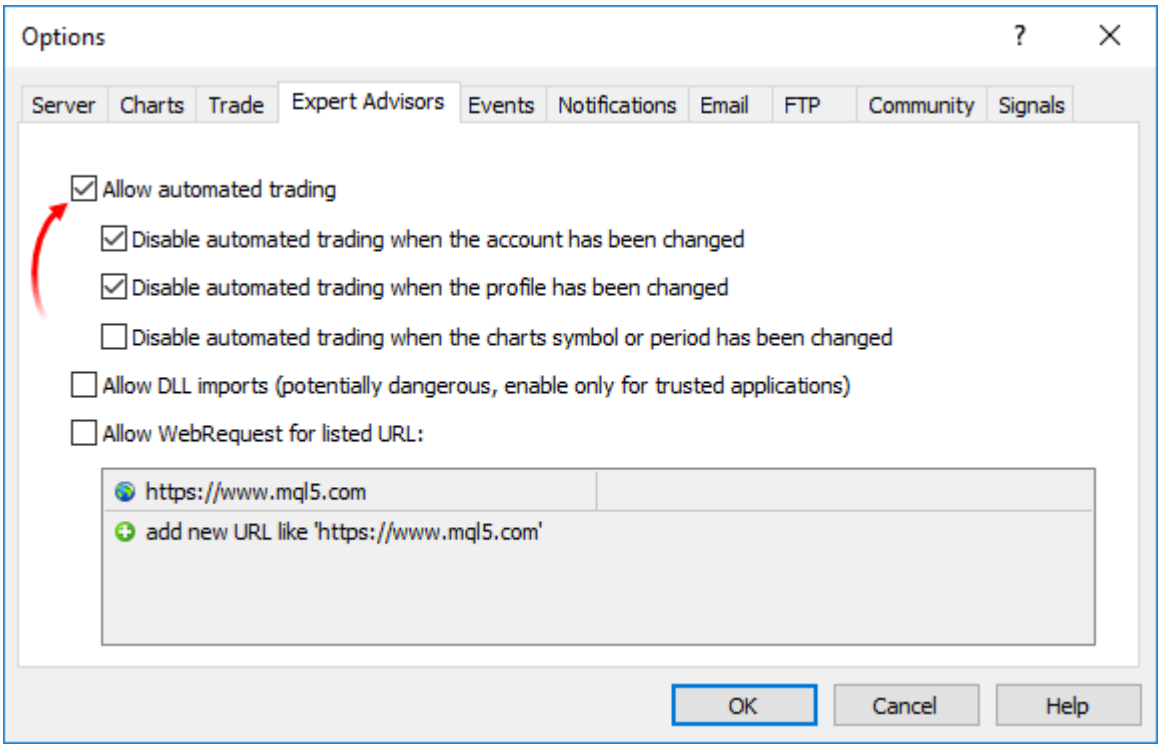
Comprobación del permiso para el trading automático

Para crear un EA seguro que realmente sea capaz de trabajar en modo automático sin la intervención del hombre, hay que organizar una serie de comprobaciones necesarias. En primer lugar, hay que comprobar de forma programática si existe realmente el permiso para realizar las operaciones comerciales. La comprobación del permiso para tradear es primordial y obligatoria durante el desarrollo de cualquier sistema automatizado.

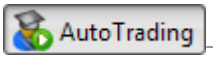
© 2000-2025, MetaQuotes Ltd.

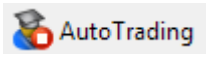
Comprobación del permiso para el trading automático en el terminal

En los ajustes del terminal se puede prohibir o permitir el trading automático para todos los programas.



También se puede usar el Panel estándar del terminal para seleccionar la opción del permiso del trading automático:

- 

- el trading automático está permitido, se puede usar las funciones comerciales en los programas iniciados.
- 

- el trading automático está prohibido, los programas iniciados van a funcionar pero las funciones comerciales no van a ejecutarse.

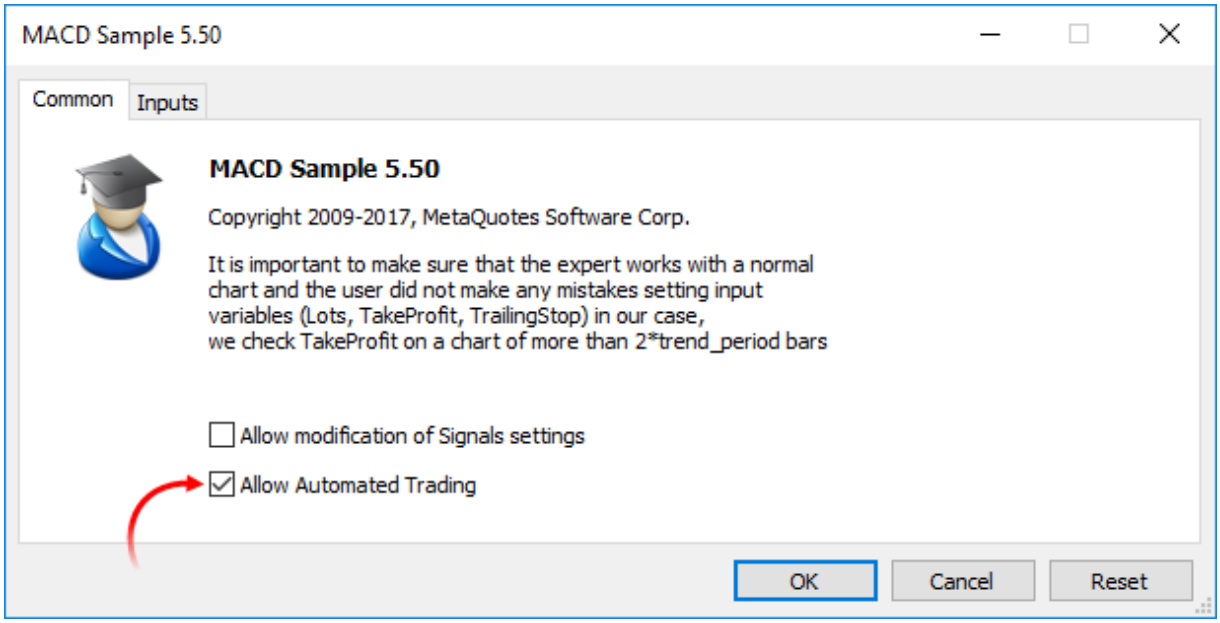
Ejemplo de comprobación:

```
if (!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("¡Compruebe el permiso para el trading automático en los ajustes del terminal
```

Comprobación del permiso para tradear para este EA/script iniciado

Al iniciar un programa, podemos permitir o prohibir el trading automático concretamente para este programa. Para eso existe un ajuste especial en las propiedades del programa.

© 2000-2025, MetaQuotes Ltd.



Ejemplo de comprobación:

```
if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("¡Compruebe el permiso para el trading automático en los ajustes del termi
else
  {
   if(!MQLInfoInteger(MQL_TRADE_ALLOWED))
      Alert("El trading automático está prohibido en las propiedades del programa p
  }
```

Comprobación del permiso para tradear con cualquier EAs/scripts para esta cuenta

La prohibición para el trading automático puede establecerse en el lado del servidor comercial. Ejemplo de comprobación de esta situación:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_EXPERT))
   Alert("El trading automático está prohibido para la cuenta ",AccountInfoInteger
   " en el lado del servidor");
```

Si el trading automático está prohibido para la cuenta comercial, las operaciones comerciales de los EAs/scripts no van a ejecutarse.

Comprobación del permiso para tradear para esta cuenta

Hay ocasiones cuando para una determinada cuenta está prohibida cualquier operación comercial, es decir, no se puede tradear ni manualmente ni con los EAs. Ejemplo cuando a la cuenta comercial se han conectado usando la contraseña de inversor:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
   Comment("El trading está prohibido para la cuenta ",AccountInfoInteger(ACCOUNT_I
         ".\n Puede que la conexión a la cuenta ha sido realizada por la contraseña
         "\n Compruebe si en el diario del terminal figura esta entrada:",
```

© 2000-2025, MetaQuotes Ltd.

```
            "\n\'",AccountInfoInteger(ACCOUNT_LOGIN),"\': trading has been disabled -
```

AccountInfoInteger(ACCOUNT_TRADE_ALLOWED) puede devolver false en la siguientes situaciones:

- no hay conexión con el servidor. Se puede comprobar usando TerminalInfoInteger(TERMINAL_CONNECTED);
- la cuenta ha sido pasada en el modo read-only (enviada al archivo);
- el trading está prohibido en el lado del servidor comercial;
- para la conexión a la cuenta se ha utilizado el modo de inversor.

Véase también

Estado del terminal de cliente, Información sobre la cuenta, Información sobre el programa MQL5-en ejecución

© 2000-2025, MetaQuotes Ltd.

Eventos de terminal de cliente

Init

Inmediatamente después de que el terminal de cliente cargue el programa (Asesor Experto o indicador personalizado) y arranque el proceso de inicialización de las variables globales, se enviará el evento Init que se maneja con la función OnInit() (si hay). Este evento también se genera después del cambio del instrumento financiero y/o período del gráfico, después de recompilación del programa en MetaEditor, después del cambio de parámetros de entrada desde la ventana de ajustes de Asesor Experto o indicador personalizado. Un Asesor Experto también se inicializa después del cambio de cuenta. El evento init no se genera para los scripts.

Deinit

Antes de que las variables globales se deinicialicen y el programa (Asesor Experto o indicador personalizado) se descargue, el terminal de cliente envía el evento Deinit al programa. El evento Deinit también se genera cuando el terminal de cliente finaliza su trabajo, cuando se cierra el gráfico, justo antes del cambio del instrumento financiero y/o período del gráfico, si el programa ha sido recompilado con éxito, con el cambio de parámetros de entrada, y con el cambio de cuenta.

Se puede obtener la razón de deinicialización del parámetro que ha sido pasado a la función OnDeinit(). La ejecución de la función OnDeinit() se limita con dos segundos y medio. Si la función no se ha completado en este este tiempo, su ejecución se termina de una manera forzosa. El evento Deinit no se genera para los scripts.

Start

El evento Start es un evento especial para activar un script o servicio después de cargarlo. Este evento es procesado por la función OnStart. El evento Start no se manda a los Asesores Expertos e indicadores personalizados.

NewTick

El evento NewTick se genera con la llegada de nuevas cotizaciones y se procesa por la función OnTick() de los Asesores Expertos adjuntos. Si con la llegada de nuevas cotizaciones la función OnTick está en ejecución para las cotizaciones anteriores, en este caso el Asesor Experto ignorará la cotización que ha llegado, porque el evento correspondiente no va a ponerse en la cola de eventos del Asesor Experto.

Todas las cotizaciones que llegan durante la ejecución del programa se ignoran hasta que se finalice la ejecución correspondiente de la función OnTick(). Después de eso, la función se iniciará sólo después de que se reciba una cotización nueva.

El evento OnTick() se genera independientemente de que si el comercio automático está permitido o no (el botón "Permitir/prohibir Auto trading").

Calculate

El evento Calculate se genera sólo para los indicadores justo después del envío del evento init y con cualquier cambio de datos de precio. Se procesa por la función OnCalculate.

Timer

© 2000-2025, MetaQuotes Ltd.

El evento Timer se genera periódicamente por el Terminal de Cliente para el Asesor Experto que ha activado el temporizador utilizando la función EventSetTimer. Habitualmente esta función se invoca en la función OnInit. El evento Timer se procesa por la función OnTimer. Una vez terminado el trabajo del Asesor Experto, hay que borrar el temporizador creado utilizando la función EventKillTimer a la que suelen llamar en la función OnDeinit.

Trade

El evento Trade se genera cuando en el servidor comercial se completa una operación comercial. La función OnTrade() maneja el evento Trade para las siguientes operaciones comerciales:

- envío, modificación o eliminación de una orden pendiente;
- cancelación de una orden pendiente por falta de medios o a la expiración del plazo de vigencia;
- activación de una orden pendiente;
- apertura, adición o cierre de una posición (o parte de posición);
- modificación de una posición abierta (cambio de stops).

TradeTransaction

Como resultado de ejecución de ciertas acciones con la cuenta de trading su estado se cambia. A estas acciones les pertenecen:

- El envío de una solicitud comercial por parte de cualquier aplicación MQL5 en el terminal de cliente utilizando la función OrderSend y OrderSendAsync, con su posterior ejecución;
- El envío de una solicitud comercial a través de la interfaz gráfica del terminal y su posterior ejecución;
- El accionamiento de órdenes pendientes y órdenes Stop en el servidor;
- La ejecución de operaciones en el servidor de trading.

Como resultado de estas acciones, para la cuenta se ejecutan las transacciones comerciales:

- tramitación de la solicitud comercial;
- cambio de órdenes abiertas;
- cambio del historial de órdenes;
- cambio del historial de operaciones;
- cambio de posiciones.

Por ejemplo, al enviar una orden de compra, ésta se tramita, para la cuenta se crea una orden de compra correspondiente, se realiza la ejecución de la orden, su eliminación de la lista de las abiertas, se agrega al historial de órdenes, luego la operación correspondiente se agrega al historial, y se crea una posición nueva. Pues todas estas acciones son transacciones comerciales. La llegada de cada una de estas transacciones al terminal es un evento TradeTransaction. Este evento se procesa con la función OnTradeTransaction.

Tester

El evento Tester se genera al terminarse el test de un Asesor Experto respecto a los datos históricos. El procesamiento del evento Tester se realiza usando la función OnTester().

TesterInit

© 2000-2025, MetaQuotes Ltd.

El evento TesterInit se genera cuando se inicia el proceso de optimización en el Probador de Estrategias antes del primer repaso. El procesamiento del evento TesterInit se realiza por la función OnTesterInit().

TesterPass

El evento TesterPass se genera cuando llega un nuevo frame de datos. El procesamiento del evento TesterPass se realiza por la función OnTesterPass().

TesterDeinit

El evento TesterDeinit se genera cuando se termina el proceso de optimización del EA en el Probador de Estrategias. El procesamiento del evento TesterDeinit se realiza por la función OnTesterDeinit().

ChartEvent

Los eventos ChartEvent se generan por el Terminal de Cliente cuando el usuario trabaja con el gráfico:

- teclazo cuando la ventana del gráfico se encuentra enfocada;
- creación de objetos gráficos;
- eliminación de objetos gráficos;
- clic con ratón en un objeto gráfico que pertenece al gráfico;
- arrastre de un objeto gráfico con ratón;
- Fin de edición del texto en LabelEdit.

Además, existe un evento de usuario ChartEvent que puede ser enviado al Asesor Experto por cualquier programa mql5 utilizando la función EventChartCustom. El evento es procesado por la función OnChartEvent.

BookEvent

El terminal de cliente genera el evento BookEvent si se cambia el estado de profundidad de mercado; este evento se procesa por la función OnBookEvent. Para que el terminal de cliente empiece a generar el evento BookEvent para un símbolo especificado, es suficiente suscribirse previamente a la recepción de estos eventos para este símbolo a través de la función MarketBookAdd.

Para dar de baja la recepción del evento BookEvent para un símbolo, es necesario llamara a la función MarketBookRelease. El evento BookEvent es de difusión, lo que significa que si un Asesor Experto se suscribe a la recepción de este evento a través de la función MarketBookAdd, todos los demás Asesores que tienen el manejador OnBookEvent van a recibir este evento. Por eso hace falta analizar el nombre del símbolo que se pasa en el manejador como un parámetro.

Véase también

Funciones de procesamiento de eventos, Ejecución de programas

© 2000-2025, MetaQuotes Ltd.

Recursos

El uso de la gráfica y sonidos en los programas MQL5

Los programas escritos en MQL5 permiten trabajar con los archivos de sonido e imágenes:

- PlaySound() reproduce un archivo de audio;
- ObjectCreate() permite crear las interfaces personalizadas utilizando los objetos gráficos OBJ_BITMAP y OBJ_BITMAP_LABEL.

PlaySound()

Un ejemplo de la llamada de la función PlaySound():

```
//+------------------------------------------------------------------+
//|  la función llama a la función estándar OrderSend() y reproduce  |
//|  el sonido                                                       |
//+------------------------------------------------------------------+
void OrderSendWithAudio(MqlTradeRequest  &request, MqlTradeResult &result)
  {
  //--- mandamos la solicitud al servidor
   OrderSend(request,result);
   //--- si la solicitud se acepta, reproducimos el sonido Ok.wav
   if(result.retcode==TRADE_RETCODE_PLACED) PlaySound("Ok.wav");
   //--- en caso del fallo, reproducimos el sonido de alarma desde el archivo timeout.
   else PlaySound("timeout.wav");
  }
```

En este ejemplo se nos muestra cómo reproducir los sonidos desde los archivos Ok.wav y timeoit.wav que entran en la entrega estándar del terminal. Estos archivos se ubican en la carpeta **directorio_del_terminal\Sounds**. Aquí el **directorio_del_terminal** significa la carpeta desde la que ha sido iniciado el terminal de cliente MetaTrader 5. La ubicación del directorio del terminal se puede averiguar desde el programa mql5 de la siguiente manera:

```
//--- La carpeta en la que guardan los datos del terminal
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

Usted puede utilizar los archivos de sonido no sólo desde la carpeta **directorio_del_terminal\Sounds**, sino también desde cualquier otra subcarpeta que se encuentra en la carpeta **directorio_del_terminal\MQL5.** La ubicación de la carpeta de datos del terminal en el ordenador se puede averiguar a través del menú del terminal "Archivo"-"Abrir carpeta de datos" o mediante el método programado:

```
//--- La carpeta en la que guardan los datos del terminal
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

Por ejemplo, si el archivo de sonido Demo.wav se encuentra en la carpeta directorio_de_datos_del_terminal\MQL5\Files, entonces la llamada de PlaySound() debe estar escrita de la siguiente manera:

```
//--- vamos a reproducir el archivo de sonido Demo.wav desde la carpeta directorio_de_
```

© 2000-2025, MetaQuotes Ltd.

```
   PlaySound("\\Files\\Demo.wav");
```

Preste la atención a que en el comentario la ruta del archivo está escrita con el símbolo "\", mientras que en la misma función se utiliza la secuencia "\\" para separar las carpetas dentro de la ruta.

A la hora de indicar una ruta siempre use sólo doble barra diagonal inversa como separador, puesto que una barra inversa solitaria es un símbolo de control para el compilador durante el análisis de las cadenas constantes y constantes de caracteres en el código fuente del programa.

Para parar la reproducción del archivo, hay que llamar a la función PlaySound() con el parámetro NULL:

```
//--- la llamada a PlaySound() con el parámetro NULL para la reproducción del sonido
   PlaySound(NULL);
```

ObjectCreate()

Ejemplo del EA que a través de la función ObjectCreate() crea el objeto "Etiqueta gráfica" (OBJ_BITMAP_LABEL).

```
string label_name="currency_label";         // nombre del objeto OBJ_BITMAP_LABEL
string euro      ="\\Images\\euro.bmp";     // ruta del archivo directorio_de_datos_del
string dollar    ="\\Images\\dollar.bmp";   // ruta del archivo directorio_de_datos_del
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- creamos el botón OBJ_BITMAP_LABEL si no lo tenemos todavía
   if(ObjectFind(0,label_name)<0)
     {
      //--- intentaremos crear el objeto OBJ_BITMAP_LABEL
      bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
      if(created)
        {
         //--- enlazamos el botón a la esquina superior derecha del gráfico
         ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
         //--- ahora ajustaremos las propiedades del objeto
         ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
         ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
         //--- pondremos a 0 el código del último error
         ResetLastError();
         //--- cargaremos la imagen para el estado del botón "Pulsado"
         bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
         //--- comprobaremos el resultado
         if(!set)
           {
            PrintFormat("Fallo al cargar la imagen desde el archivo %s. Código del er
           }
         ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- cargaremos la imagen para el estado del botón "Despulsado"
         set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);

         if(!set)
           {
            PrintFormat("Fallo al cargar la imagen desde el archivo %s. Código del er
           }
         //--- enviaremos al gráfico el comando de reinicio para que el botón aparezc
         ChartRedraw(0);
        }
      else
        {
         //--- fallo al crear el objeto, avisaremos sobre ello
         PrintFormat("Fallo al crear el objeto OBJ_BITMAP_LABEL. Código del error %d",
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- quitaremos el objeto desde el gráfico
   ObjectDelete(0,label_name);
  }
```

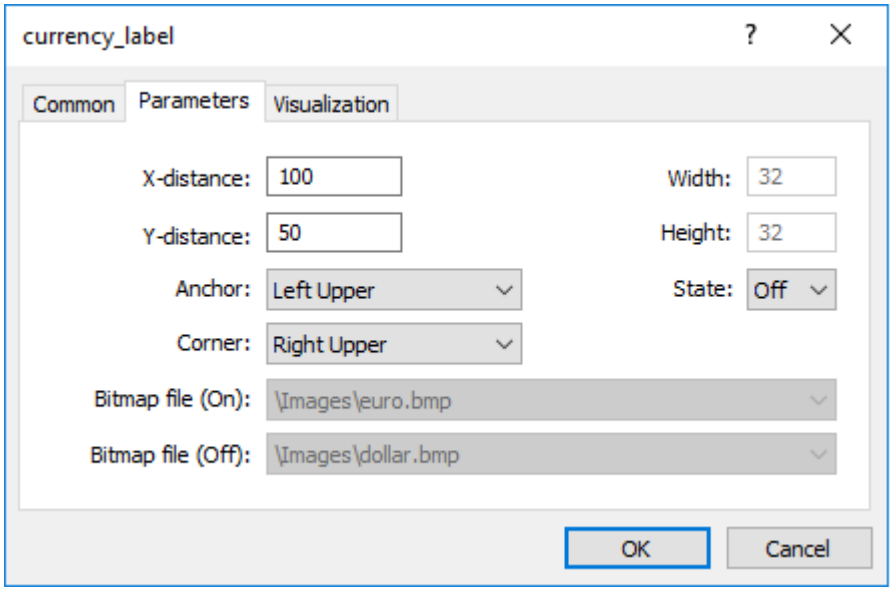
La creación y configuración del objeto gráfico con el nombre currency_label se llevan a cabo en la función OnInit(). Las rutas hacia los archivos de imágenes se establecen en las variables globales **euro** y **dollar**, como separador se utiliza la barra diagonal inversa doble:

```
string euro      ="\\Images\\euro.bmp";    // ruta del archivo directorio_de_datos_del
string dollar    ="\\Images\\dollar.bmp";  // ruta del archivo directorio_de_datos_del
```

En este caso los archivos se encuentran en la carpeta **directorio_de_datos_del_terminal**\MQL5\Images.

En realidad el objeto OBJ_BITMAP_LABEL representa un botón que en función de su estado (pulsado o despulsado) puede visualizar una de las dos imágenes: euro.bmp o dollar.bmp.

© 2000-2025, MetaQuotes Ltd.



El tamaño del botón con la interfaz gráfica se ajusta automáticamente al tamaño de la imagen a mostrar. La imagen se cambia con el clic izquierdo en el objeto OBJ_BITMAP_LABEL (**en las propiedades tiene que estar seleccionada la opción "Desactivar la selección"**). El objeto OBJ_BITMAP se crea de la misma manera y sirve para crear el fondo con la imagen necesaria.

El valor de la propiedad OBJPROP_BMPFILE que responde de la apariencia de los objetos OBJ_BITMAP y OBJ_BITMAP_LABEL se puede cambiar de forma dinámica. Esto permite crear diferentes interfaces personalizadas interactivas para los programas mql5.

Inserción de recursos en los archivos ejecutables durante la compilación de programas mql5

Un programa mql5 a lo mejor podrá necesitar varios diferentes recursos cargables en forma de archivos de imágenes o sonidos. Para evitar la necesidad de transferir todos estos archivos durante el movimiento del programa ejecutable en MQL5, se debe utilizar la directiva #resource:

```
#resource ruta_hacia_archivo_del_recurso
```

El comando #resource indica al compilador que hay que hay que incluir el recurso según la ruta especificada **ruta_hacia_archivo_del_recurso** en el archivo ejecutable EX5. De esta manera, se puede colocar todas las imágenes y sonidos necesarios directamente en el archivo EX5, sin tener que pasar todos los archivos que utiliza el programa para que funcione en otro terminal. Cualquier archivo EX5 puede contener recursos, y cualquier programa EX5 puede utilizar los recursos desde otro programa EX5.

Los archivos en el formato BMP y WAV se comprimen automáticamente antes de ser insertados en el archivo ejecutable EX5. Esto quiere decir que el uso de los recursos no sólo permite crear los programas MQL5 de pleno valor sino también reduce el tamaño total de los archivos requeridos por el terminal a la hora de utilizar la gráfica y el audio en comparación con el modo común de creación de los programas mql5.

El tamaño del archivo de un recurso no puede superar 128 Mb.

© 2000-2025, MetaQuotes Ltd.

Búsqueda de recursos especificados por el compilador

Un recurso puede ser insertado mediante el comando #resource "<ruta_hacia_archivo_del_recurso>"

```
#resource "<ruta_hacia_archivo_del_recurso>"
```

La longitud de la cadena constante <ruta_hacia_archivo_del_recurso> no puede ser más de 63 caracteres.

El compilador busca el recurso especificado en orden siguiente:

- si la ruta se empieza con la barra inversa separadora "\" (se escribe "\\"), entonces el recurso se busca respecto al catálogo **carpeta_de_datos_del_terminal**\MQL5\,
- si no hay ninguna barra inversa, el recurso se busca respecto a la ubicación del archivo fuente en el que este recurso ha sido insertado.

En la ruta del recurso no se puede utilizar las subcadenas "..\\" y ":\\".

Algunos ejemplos de inserción de recursos:

```
//--- la especificación correcta de los recursos
#resource "\\Images\\euro.bmp" // euro.bmp se encuentra en carpeta_de_datos_del_termi
#resource "picture.bmp"        // picture.bmp se encuentra en la misma carpeta que el
#resource "Resource\\map.bmp"  // recurso se encuentra en la carpeta catálogo_del_arch

//--- indicación incorrecta de los recursos
#resource ":picture_2.bmp"     // no se puede utilizar ":"
#resource "..\\picture_3.bmp"  // no se puede utilizar ".."
#resource "\\Files\\Images\\Folder_First\\My_panel\\Labels\\too_long_path.bmp" //más c
```

Uso de recursos

Nombre del recurso

Después de que el recurso haya sido declarado mediante la directiva #resource, puede utilizarlo en cualquier parte del programa. El nombre del recurso será su ruta sin la barra inversa al principio de la línea que establece la ruta del archivo. Para poder utilizar su propio recurso en el código, hay que añadir el signo especial "::" antes del nombre de este recurso.

Ejemplos:

```
//--- ejemplos de indicación de recursos y sus nombres en el comentario
#resource "\\Images\\euro.bmp"          // nombre del recurso - Images\euro.bmp
#resource "picture.bmp"                 // nombre del recurso - picture.bmp
#resource "Resource\\map.bmp"           // nombre del recurso - Resource\map.bmp
#resource "\\Files\\Pictures\\good.bmp" // nombre del recurso - Files\Pictures\good.br
#resource "\\Files\\Demo.wav";          // nombre del recurso - Files\Demo.wav"
#resource "\\Sounds\\thrill.wav";       // nombre del recurso - Sounds\thrill.wav"
...
```

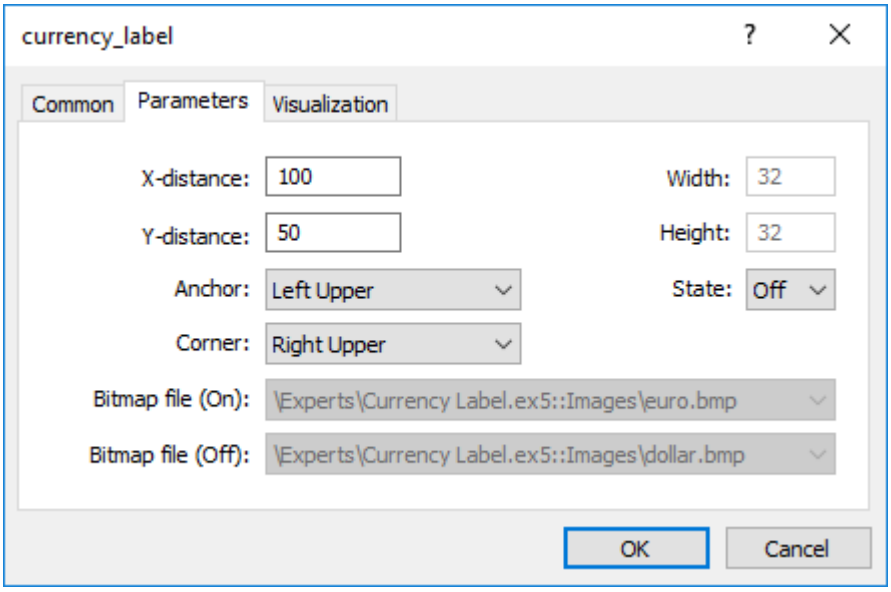
© 2000-2025, MetaQuotes Ltd.

```
//--- uso de recursos
ObjectSetString(0,bitmap_name,OBJPROP_BMPFILE,0,"::Images\\euro.bmp");
...
ObjectSetString(0,my_bitmap,OBJPROP_BMPFILE,0,"::picture.bmp");
...
set=ObjectSetString(0,bitmap_label,OBJPROP_BMPFILE,1,"::Files\\Pictures\\good.bmp");
...
PlaySound("::Files\\Demo.wav");
...
PlaySound("::Sounds\\thrill.wav");
```

Cabe mencionar que cuando se establece una imagen desde el recurso para los objetos OBJ_BITMAP y OBJ_BITMAP_LABEL, el valor de la propiedad OBJPROP_BMPFILE ya no se puede cambiar manualmente. Por ejemplo, estamos utilizando los recursos de los archivos euro.bmp y dollar.bmp para crear OBJ_BITMAP_LABEL.

```
#resource "\\Images\\euro.bmp";    // euro.bmp se encuentra en carpeta_de_datos_del_te
#resource "\\Images\\dollar.bmp";  // dollar.bmp se encuentra en carpeta_de_datos_del_
```

Entonces si nos fijamos en las propiedades de este objeto, veremos que las propiedades BitMap File (On) y BitMap File (Off) tienen el color gris y no están disponibles para el cambio manual:



Uso de recursos de otros programas mql5

El uso de los recursos también tiene otra ventaja - en cualquier programa mql5 se puede utilizar los recursos desde cualquier archivo EX5. De esta manera, los recursos desde un archivo EX5 se puede utilizar en muchos otros programas mql5.

Para poder usar el nombre del recurso desde otro archivo, hay que indicarlo como sigue <ruta_nombre_del_archivo_EX5>::<nombre_del_recurso>. Por ejemplo, supongamos que el script Draw_Triangles_Script.mq5 contiene el recurso para una imagen en el archivo triangle.bmp:

© 2000-2025, MetaQuotes Ltd.

```
#resource "\\Files\\triangle.bmp"
```

Entonces su nombre para el uso en el mismo script será el siguiente "Files\triangle.bmp", y para poder usarlo hay que añadir a su nombre el signo especial "::".

```
//--- el uso del recurso en el mismo script
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

Para tener la posibilidad de usar el mismo recurso desde otro programa, por ejemplo desde un Asesor Experto, hay que añadir al nombre del recurso la ruta del archivo EX5 respecto a la carpeta **carpeta_de_datos_del_terminal\MQL5** y el nombre del archivo EX5 de este script - **Draw_Triangles_Script.ex5**. Supongamos que el script se encuentra en la carpeta estándar **carpeta_de_datos_del_terminal\MQL5**\Scripts\, entonces la llamada hay que escribir de la siguiente manera:

```
//--- el uso del recurso del script en el Asesor Experto
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\\Draw_Triangles_Script.e
```

Si durante la llamada al recurso de otro archivo EX5 no indicamos la ruta de este archivo ejecutable, entonces la búsqueda de este archivo ejecutable se realiza en la misma carpeta donde se encuentra el programa que ha llamado al recurso. Eso quiere decir que si en el Asesor Experto se llama al recurso desde el archivo Draw_Triangles_Script.ex5 sin especificar la ruta, por ejemplo así:

```
//--- llamada al recurso del script en el EA sin especificar la ruta
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files\\
```

entonces el archivo va a buscarse en la carpeta **carpeta_de_datos_del_terminal\MQL5**\Experts\ si el mismo EA se encuentra en la carpeta **carpeta_de_datos_del_terminal\MQL5**\Experts\.

Trabajo con los indicadores personalizados incluidos como recursos

Para el funcionamiento de los programas mql5 puede ser necesario uno o varios indicadores personalizados. Todos ellos pueden incluirse en el código del programa mql5 a ejecutar. La inclusión de los indicadores como recursos permite facilitar la distribución de la aplicación.

A continuación se muestra el ejemplo de inclusión y uso del indicador SampleIndicator.ex5 ubicado en la carpeta: **directorio_de_datos_del_terminal**\MQL5\Indicators\:

```
//+------------------------------------------------------------------+
//|                                                     SampleEA.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#resource "\\Indicators\\SampleIndicator.ex5"
int handle_ind;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   handle_ind=iCustom(_Symbol,_Period,"::Indicators\\SampleIndicator.ex5");
   if(handle_ind==INVALID_HANDLE)
     {
      Print("Expert: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
//--- ...
   return(INIT_SUCCEEDED);
  }
```

El caso cuando un indicador personalizado crea en la función OnInit() una o varias copias de sí mismo requiere un análisis más detenido. Recordemos que para el uso del recurso desde el programa mql5, hay que indicarlo como sigue: **<ruta_nombre_del_archivo_EX5>::<nombre_del_recurso>**.

Por ejemplo, si el indicador SampleIndicator.ex5 se incluye como recurso en el Asesor Experto SampleEA.ex5, la ruta hacia sí mismo especificada durante la llamada de iCustom() en la función de inicialización del indicador personalizado, tendrá la siguiente apariencia: "\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5". Si esta ruta se establece de forma explícita, el indicador personalizado SampleIndicator.ex5 será vinculado rígidamente al Asesor Experto SampleEA.ex5 y perderá la capacidad de actuar independientemente.

La ruta hacia sí mismo se puede conseguir a través de la función GetRelativeProgramPath(), su ejemplo se muestra más abajo:

```
//+------------------------------------------------------------------+
//|                                              SampleIndicator.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property indicator_separate_window
#property indicator_plots 0
int handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- modo incorrecto de indicar el vínculo hacia sí mismo
//--- string path="\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5";
//--- modo correcto de obtener el vínculo hacia sí mismo
  string path=GetRelativeProgramPath();
//--- indicator buffers mapping
   handle=iCustom(_Symbol,_Period,path,0,0);
   if(handle==INVALID_HANDLE)
     {
      Print("Indicator: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
   else Print("Indicator handle=",handle);
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
///....
//+------------------------------------------------------------------+
//| GetRelativeProgramPath                                           |
//+------------------------------------------------------------------+
string GetRelativeProgramPath()
  {
   int pos2;
//--- obtenemos la ruta absoluta hacia la aplicación
   string path=MQLInfoString(MQL_PROGRAM_PATH);
//--- buscamos la posición de la subcadena "\MQL5\"
   int    pos =StringFind(path,"\\MQL5\\");
//--- subcadena no encontrada - error
   if(pos<0)
      return(NULL);
//--- saltamos la carpeta "\MQL5"
   pos+=5;
//--- saltamos los símbolos '\' que sobran
   while(StringGetCharacter(path,pos+1)=='\\')
      pos++;
//--- si es un recurso, devolvemos la ruta respecto al directorio MQL5
   if(StringFind(path,"::",pos)>=0)
      return(StringSubstr(path,pos));
//--- buscamos el divisor para el primer subdirectorio en MQL5 (por ejemplo, MQL5\Ind
//--- si no hay, devolvemos la ruta respecto al directorio MQL5
   if((pos2=StringFind(path,"\\",pos+1))<0)
      return(StringSubstr(path,pos));
//--- devolvemos la ruta respecto al subdirectorio (por ejemplo, MQL5\Indicators)
   return(StringSubstr(path,pos2+1));
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double& price[])
  {
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Variables de recurso

© 2000-2025, MetaQuotes Ltd.

Los recursos se pueden declarar con la ayuda de variables de recurso y llamarse como si fueran una variable del tipo correspondiente. Formato de la declaración:

```
#resource ruta_al_archivo_del_recurso as tipo_de_variable_de_recurso nombre_de_la_vari
```

Ejemplos de declaración:

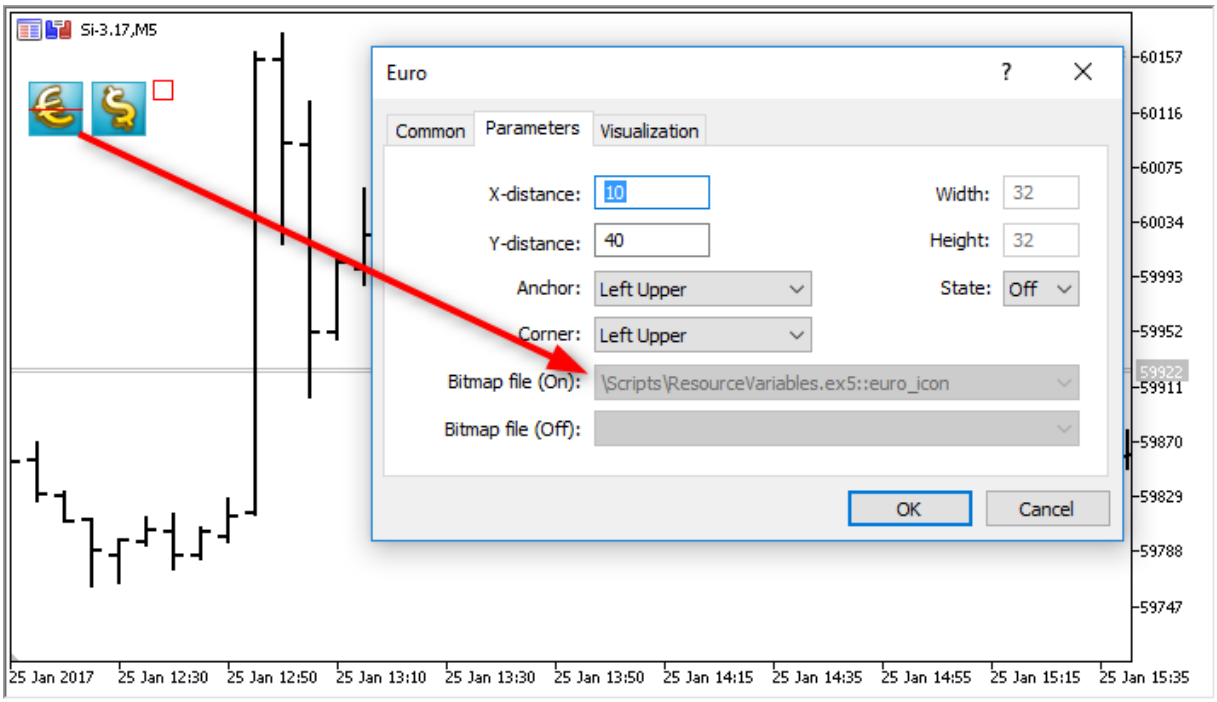
```
#resource "data.bin" as int ExtData[]              // declaración de una matriz de tipo
#resource "data.bin" as MqlRates ExtData[]         // declaración de una matriz de esti
//--- líneas
#resource "data.txt" as string ExtCode             // declaración de la línea que conti
//--- recursos gráficos
#resource "image.bmp" as bitmap ExtBitmap[]        // declaración de la matriz que cont
#resource "image.bmp" as bitmap ExtBitmap2[][]     // declaración de la matriz bidimens
```

Con tal declaración solo es posible dirigirse a los datos del recurso a través de una variable, el direccionamiento automático a través de "::<rsource name>" no funciona.

```
#resource "\\Images\\euro.bmp" as bitmap euro[][]
#resource "\\Images\\dollar.bmp"
//+------------------------------------------------------------------+
//|  Función de creación del objeto OBJ_BITMAP_LABEL con la ayuda del recurso     |
//+------------------------------------------------------------------+
void Image(string name,string rc,int x,int y)
  {
   ObjectCreate(0,name,OBJ_BITMAP_LABEL,0,0,0);
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
   ObjectSetString(0,name,OBJPROP_BMPFILE,rc);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- introduciomos el tamaño de la imagen [width, height], que se guarda en la variab
   Print(ArrayRange(euro,1),", ",ArrayRange(euro,0));
//--- cambiamos la imagen a euro - dibujamos una franja horizontal roja en la mitad
   for(int x=0;x<ArrayRange(euro,1);x++)
      euro[ArrayRange(euro,1)/2][x]=0xFFFF0000;
//--- creamos el recurso gráfico con la ayuda de la variable de recurso
   ResourceCreate("euro_icon",euro,ArrayRange(euro,1),ArrayRange(euro,0),0,0,ArrayRang
//--- creamos el objeto marca gráfica Euro, al que le colocamos la imagen del recurso
   Image("Euro","::euro_icon",10,40);
//--- otro método de utilización del recurso, no podemos dibujar en él
   Image("USD","::Images\\dollar.bmp",15+ArrayRange(euro,1),40);
//--- el método directo de direccionamiento al recurso euro.bmp no está disponible, pu
   Image("E2","::Images\\euro.bmp",20+ArrayRange(euro,1)*2,40); // habrá un error de t
  }
```

© 2000-2025, MetaQuotes Ltd.

Resultado de ejecución del script - se han creado solo dos objetos OBJ_BITMAP_LABEL de tres. Además, en la imagen del primer objeto veremos una franja roja en la mitad.



Una ventaja importante del uso de recursos es que los archivos de recurso, antes de ser incluidos en el archivo ejecutable EX5, se comprimen de forma automática antes de la compilación. De esta forma, el uso de las variables de recurso permite no solo empaquetar los datos necesarios para el trabajo directamente en un archivo ejectutable EX5, sino también reducir el número y el tamaño total de los archivos en comparación con el método habitual de escritura de programas mql5.

El uso de variables de recurso es especialmente cómodo para publicar productos en el Mercado.

Particularidades

- El tipo especial de variable de recurso *bitmap* indica al compilador que el recurso es una imagen gráfica. Estas variables reciben el tipo uint.
- La matriz-variable de recurso del tipo *bitmap* puede tener dos dimensiones, en este caso, el tamaño de la matriz se definirá como [altura_de_la_imagen][anchura_de_la_imagen]. En el caso de que la matriz sea unidimensional, el número de elementos será igual al producto altura_de_la_imagen*anchura_de_la_imagen.
- Al cargar una imagen de 24 bits para todos los píxeles de la imagen del componente del canal alfa se establece en el valor 255.
- Al cargar una imagen de 32 bits sin canal alfa, asimismo, para todos los píxeles de la imagen del componente del canal alfa se establece en el valor 255.
- Al cargar una imagen de 32 bits con canal alfa, no tiene lugar ninguna manipulación con los píxeles.
- El tamaño del archivo del recurso no puede ser superior a 128 Mb.
- Para los archivos de línea se define la codificación de forma automática por la presencia de BOM (encabezamiento). Si no hay BOM, la codificación se determina por el contenido del archivo. Tienen soporte los archivos en la codificación ANSI, UTF-8 y UTF-16. Al leer los datos de los archivos, todas las líneas se transforman en Unicode.

© 2000-2025, MetaQuotes Ltd.

Programas en OpenCL

El uso de variables de línea de recurso puede facilitar significativamente la escritura de algunos programas. Por ejemplo, podrá escribir el código de un programa OpenCL en un archivo CL aparte, y después incluir este archivo en forma de línea en los recursos de su programa MQL5.

```
#resource "seascape.cl" as string cl_program
...
int context;
if((cl_program=CLProgramCreate(context,cl_program)!=INVALID_HANDLE)
  {
   //--- ejecutamos las acciones posteriores con el programa OpenCL
  }
```

En este ejemplo, sin uso de la variable de recurso *cl_program*, usted tendría que describir todo el código en forma de una gran variable de línea.

Véase también

ResourceCreate(), ResourceSave(), PlaySound(), ObjectSetInteger(), ChartApplyTemplate(), Operaciones con archivos

© 2000-2025, MetaQuotes Ltd.

Llamadas a las funciones importadas

Para importar funciones durante la ejecución de un programa mql5 se utiliza la ligadura temprana. Eso significa que si en el programa hay llamada a una función importada, el módulo correspondiente (ex5 o dll) se carga durante la carga del programa. Las bibliotecas MQL5 y DLL se ejecutan en el flujo del módulo-invocador.

No se recomienda utilizar el nombre completamente especificado del módulo a cargar como el *Drive:\Directory\FileName.Ext*. Las bibliotecas MQL5 se cargan de la carpeta *terminal_dir\MQL5\Libraries*. Si no encontramos ninguna, intentamos cargar la biblioteca desde la carpeta *terminal_dir\experts*.

Las bibliotecas de sistema (DLL) se cargan según las reglas del sistema operativo. Si la biblioteca ya está cargada (por ejemplo, por otro Asesor Experto, e incluso de otro terminal de cliente inicializado en paralelo), entonces nos dirigimos a la biblioteca ya cargada. En caso contrario, la búsqueda se efectua como sigue:

1. Directorio del que ha sido iniciado el módulo que importa dll. Hablando del módulo, se quiere decir un Asesor Experto, un script, un indicador o biblioteca EX5;
2. Directorio directorio_de_terminal_datos\MQL5\Libraries (TERMINAL_DATA_PATH\MQL5\Libraries);
3. Directorio del que ha sido iniciado el terminal de cliente de MetaTrader 5;
4. Directorio de sistema;
5. Directorio Windows;
6. Directorio corriente;
7. Directorios enumerados en la variable de sistema PATH.

Si una biblioteca DLL utiliza en su trabajo otra biblioteca, la primera no puede ser cargada si falta la segunda DLL.

Antes de que se cargue un Asesor Experto (script, indicador) se forma una lista general de todos los módulos bibliotecarios EX5. Se prevé usarlos como del Asesor Experto cargado (script, indicador), tanto de las bibliotecas de esta lista. De esta manera cargándose sólo una vez, los módulos bibliotecarios EX5 se utilizan varias veces. Las bibliotecas usan las variables predefinidas del Asesor Experto (script, indicador) que las ha invocado.

El orden de búsqueda de la biblioteca EX5 importada es el siguiente:

1. Directorio, ruta al cual se establece con relación al directorio del Asesor Experto (script, indicador) que importa EX5;
2. Directorio directorio_de_terminal\MQL5\Libraries;
3. Directorio MQL5\Libraries en el directorio general de todos los terminales de cliente de MetaTrader 5(Common\MQL5\Libraries).

Las funciones importadas de DLL en el programa mql5 deben asegurar el acuerdo de enlaces para las funciones Windows API. Para asegurar dicho acuerdo en el texto fuente de programas escritos en C o C++ se utiliza la palabra clave __stdcall, bastante específica para los compiladores de Microsoft(r). El acuerdo en cuestión se caracteriza por lo siguiente:

- función que invoca (en nuestro caso es el programa mql5) tiene que "ver" el prototipo de función invocada (importada de DLL) para colocar los parámetros en la pila de una forma correcta;

© 2000-2025, MetaQuotes Ltd.

- función que invoca (en nuestro caso es un programa mql5) coloca los parámetros en la pila de una forma inversa - de derecha a izquierda; precisamente en este orden una función importada lee los parámetros traspasados para ella;
- parámetros se traspasan por valor, salvo los que se pasan explícitamente por referencia (en nuestro caso, líneas);
- función importada limpia la pila cuando lee losparámetros pasados para ella.

A la hora de describir el prototipo de una función importada se puede utilizar los parámetros con valores por defecto.

Si la biblioteca correspondiente no ha podido cargarse o si está prohibido usar DLL, o no se ha encontrado la función importada, entonces el Asesor Experto deja de funcionar dejando el mensaje "expert stopped" en el registrador de datos. El Asesor Experto se cargará hasta que no vuelva a ser inicializado. El Asesor Experto podrá ser reinicializado después de recompilación o después de que abramos la tabla de propiedades de Asesor y apretemos el botón "Ok".

Traspaso de parámetros

Todos los parámetros de tipos simples se traspasan por valor, si no se indica explícitamente que se pasan por referencia. Cuando se traspasa la cadena, se traspasa la dirección del buffer de la cadena copiada; si la cadena se pasa por referencia , en la función importada de DLL se pasa la dirección del buffer precisamente de esta cadena sin copiar.

Las estructuras que contienen los arrays dinámicos, cadenas, clases, otras estructuras complejas, así como los arrays dinámicos o estáticos de los objetos mencionados, no pueden ser traspasadas en la función importada en calidad de parámetros.

Cuando se traspasa un array en DLL, siempre (haya o no haya la bandera AS_SERIES) se pasa la dirección del inicio del búfer de datos. La función dentro de DLL no sabe nada de la bandera AS_SERIES, el array traspasado es un array estático de una longitud desconocida, y para especificar el tamaño del array se utiliza un parámetro adicional.

© 2000-2025, MetaQuotes Ltd.

Errores de ejecución

En el subsistema ejecutivo del terminal de cliente existe posibilidad de guardar el código de error en caso de que éste surja durante la ejecución del programa mql5. Hay una variable predefinida _LastError para cada programa mql5 ejecutable.

La variable _LastError se pone a cero antes del inicio de la función OnInit. En caso de surgir una situación errónea durante los cálculos o a la hora de llamar a una función built-in, la variable _LastError acepta el código correspondiente del error. El valor guardado en esta variable se puede obtener utilizando la función GetLastError().

Existe una serie de errores críticos, y cuando éstos surgen la ejecución del programa se detiene inmediatamente:

- división por cero;
- salida fuera de los límites del array;
- uso incorrecto de un puntero a objeto;

© 2000-2025, MetaQuotes Ltd.

Simulación de estrategias comerciales

La idea del trading automatizado es bastante atractiva con el hecho de que el robot de trading trabaja sin descanso 24 horas al día y siete días a la semana. El robot no sabe nada de cansancio, dudas y miedo, ni tampoco de problemas psicológicos. Sólo basta con formalizar las reglas de trading e implementarlas en forma de algoritmos, y su robot está listo a trabajar sin parar. Pero antes, es necesario asegurarse del cumplimiento de dos condiciones importantes:

- el Asesor Experto realiza las operaciones comerciales de acuerdo con las reglas del sistema de trading;
- la estrategia de trading que ha sido implementada en el Asesor Experto muestra la ganancia a base de los datos históricos.

Para recibir respuestas a estas preguntas, se utiliza el Probador de Estrategias que forma parte integrante del Terminal de Cliente MetaTrader 5.

En este apartado vamos a analizar todas las particularidades de la simulación y optimización de los programas en el Probador de Estrategias:

- Limitaciones del funcionamiento de las funciones en el Probador de Estrategias
- Modos de generación de ticks
- Modelación de spreads
- Uso de ticks reales durante la simulación
- Variables globales del Terminal de Cliente
- Cálculo de indicadores durante la simulación
- Carga del historial durante la simulación
- Simulación en múltiples divisas
- Modelación de la hora en el Probador
- Objetos gráficos durante la simulación
- Función OnTimer() en el Probador
- Función Sleep() en el Probador
- Uso del Probador para las tareas de optimización en los cálculos matemáticos
- Sincronización de las barras durante la simulación en el modo "Sólo precios de apertura"
- Función IndicatorRelease() en el Probador
- Procesamiento de eventos en el Probador
- Agentes de pruebas
- Intercambio de datos entre el Terminal y el Agente
- Uso de la carpeta compartida de todos los Terminales de Cliente
- Uso de la DLL

© 2000-2025, MetaQuotes Ltd.

Limitaciones de memoria y espacio en el disco en MQL5 Cloud Network

Al ejecutar la optimización en MQL5 Cloud Network, existe un límite: el asesor experto probado no puede escribir más de 4GB de datos en el disco y utilizar más de 4GB de RAM. Si se excede el límite, el agente de red no podrá completar los cálculos correctamente y usted no obtendrá el resultado de la prueba. En este caso, además, se le cobrará el tiempo ya empleado en los cálculos.

Si necesita obtener información de cada pasada de optimización, utilice el envío de frames sin escribir en el disco. En otras palabras, al realizar cálculos en MQL5 Cloud Network no utilice operaciones de archivo en los asesores durante la optimización, puede utilizar esta comprobación:.

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

Limitaciones del funcionamiento de las funciones en el Probador de Estrategias

En el Probador del terminal de cliente existen limitaciones para el trabajo de algunas funciones.

Funciones Comment(), Print() y PrintFormat()

Con el fin de aumentar la velocidad de operación durante la optimización de los parámetros del EA, las funciones Comment(), Print() y PrintFormat() no se ejecutan. La excepción es el uso de estas funciones dentro del manejador OnInit(). Eso permite facilitar la búsqueda de las causas de los errores cuando surgen.

Funciones Alert(), MessageBox(), PlaySound(), SendFTP, SendMail(), SendNotification(), WebRequest()

Las funciones de interacción con el "mundo externo" Alert(), MessageBox(), PlaySound(), SendFTP(), SendMail(), SendNotification() y WebRequest() no se ejecutan en el Probador.

Modos de generación de ticks

© 2000-2025, MetaQuotes Ltd.

Un Asesor Experto escrito en el lenguaje MQL5 es un programa que se inicia cada vez como respuesta a una acción externa - un evento. Para cada evento predefinido el EA dispone de una función correspondiente a este evento - manejador de eventos.

El evento más importante para un EA es el cambio del precio - NewTick. Por esta razón, para la simulación de los EAs es necesario generar las secuencias de ticks. En el Probador del Terminal de Cliente MetaTrader 5 hay 3 modos de generación de ticks:

- Todos los ticks
- Precios OHLC de las barras de un minuto (1 Minute OHLC)
- Sólo precios de apertura

El modo "Todos los ticks" es el modo básico y el más detallado de generación de los ticks, los dos restantes es una simplificación del modo principal y serán descritos en comparación con el modo "Todos los ticks". Vamos a analizar los tres modos para comprender la diferencia entre ellos.

Todos los ticks

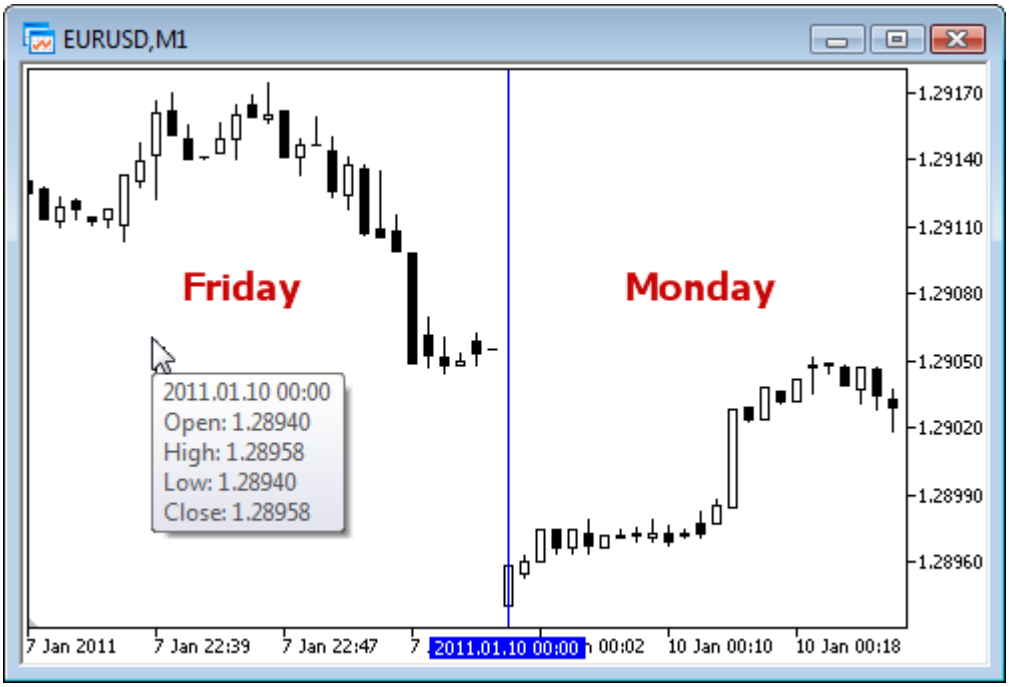
El historial de cotizaciones de los instrumentos financieros se traspasa del servidor comercial al Terminal de Cliente MetaTrader 5 en forma de los bloques de barras de un minuto bien comprimidos. Usted puede encontrar la información detallada sobre cómo se hace la solicitud y construcción de los períodos necesarios en el apartado de la ayuda Organización de acceso a los datos.

El elemento mínimo del historial de precios es una barra de un minuto desde la que se puede conseguir la información sobre los valores de cuatro precios:

- Open - precio de apertura de la barra de un minuto;
- High - el máximo alcanzado durante esta barra de un minuto;
- Low - el mínimo alcanzado durante esta barra de un minuto;
- Close - precio de cierre de la barra.

La nueva barra de un minuto no se abre en el momento cuando se empieza un nuevo minuto (el número de segundos llega a ser 0), sino cuando llega un tick nuevo, es decir, cuando el precio se cambia por lo menos en un punto. En la imagen se muestra la primera barra minutera de nueva semana de trading con fecha y hora de apertura 2011.01.10. 00:00. La diferencia de precios entre el viernes y el lunes que vemos en el gráfico es un hecho corriente, puesto que incluso durante los días de descanso las cotizaciones de divisas van cambiando en respuesta a las noticias que llegan.

© 2000-2025, MetaQuotes Ltd.

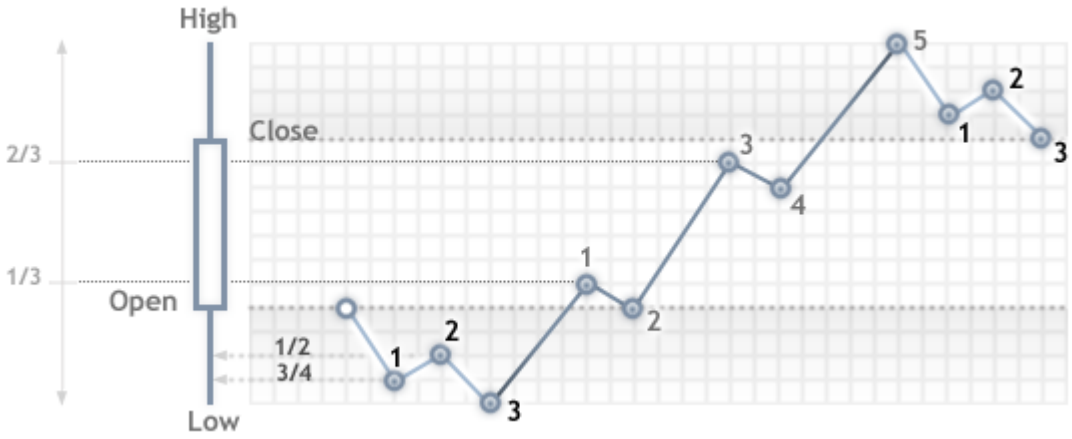


Respecto a esta barra minutera sólo sabemos que fue abierta el 10 de Enero de 2011 a las 00:00, pero no sabemos nada de los segundos. Esto podía pasar a las 00:00:12 o 00:00:36 (pasados 12 o 36 segundos desde el inicio de la nueva jornada), o cualquier otro momento dentro de este minuto. Pero lo que sabemos exactamente es que en el momento de apertura de la nueva barra minutera el precio Open para EURUSD se encontraba en 1.28940.

Igualmente, tampoco sabemos con una precisión de un segundo cuándo ha llegado el tick correspondiente al precio de cierre de la barra en cuestión. Lo único que sabemos es que se trata del último precio en esta barra minutera que ha sido apuntado como el precio Close. Para este minuto este precio es 1.28958. El tiempo de aparición de los precios High y Low tampoco se sabe. Pero sabemos que el precio máximo y el mínimo ha alcanzado sin duda alguna los niveles 1.28958 y 1.28940, respectivamente.

Para probar la estrategia comercial nos hace falta una secuencia de ticks sobre la que va a emularse el trabajo del EA. De esta manera, para cada barra de un minuto sabemos **4 puntos de control** sobre los que podemos decir con total seguridad que el precio ha estado ahí. Si una barra tiene sólo 4 ticks, esta información será suficiente para la simulación, pero normalmente el volumen de tick es superior a 4. Eso significa que hace falta generar los puntos de control adicionales para los ticks que han llegado entre los precios Open, High, Low y Close. El principio de generación de los ticks en el modo "Todos los ticks" se describe en el artículo Algoritmo de generación de los ticks en el Probador de Estrategias del terminal MetaTrader 5 la ilustración desde el cual se muestra más abajo.

© 2000-2025, MetaQuotes Ltd.



Durante la simulación el modo "Todos los ticks" la función OnTick() del EA va a **llamarse en cada punto de control**, siendo cada punto de control un tick desde la secuencia generada. El EA va a recibir la hora y el precio del tick modelado igualmente como durante el trabajo en tiempo real.

> **Importante:** el modo de simulación "Todos los ticks" es el más preciso pero también es el más duradero. Para una valoración primaria de la mayoría de las estrategias comerciales generalmente es suficiente utilizar alguno de otros dos modos de prueba.

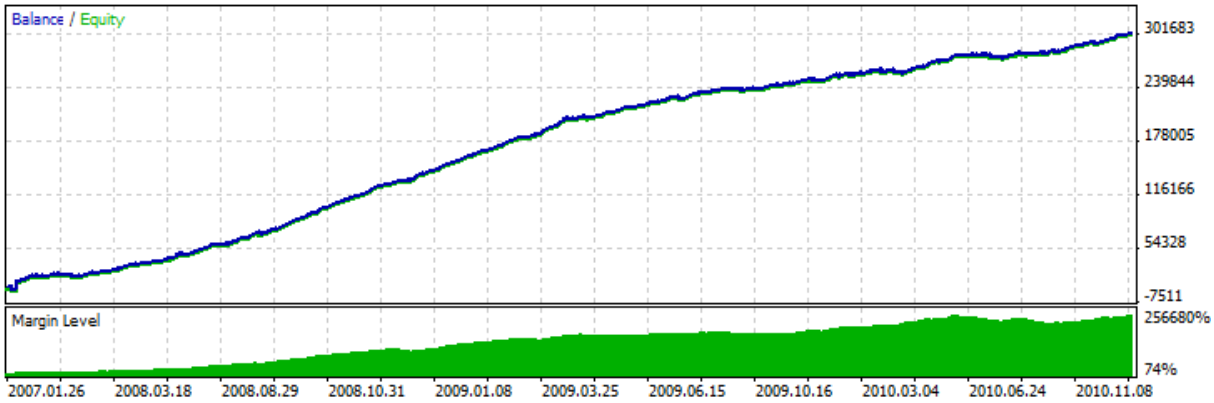
1 minute OHLC

El modo de simulación "Todos los ticks" es el más preciso de los tres, pero al mismo tiempo es el más lento. La función OnTick() se inicia para cada tick, y el volumen de tick puede ser bastante grande. Para las estrategias a las que no les importa en qué secuencia de ticks se desarrollaba el precio en el transcurso de cada barra existe el modo de modelación más rápido y más aproximado - "1 minute OHLC".

En el modo "1 minute OHLC" la secuencia de ticks se construye sólo a base de los **precios OHLC de las barras de un minuto**, en este caso el número de puntos de control generados se reduce sustancialmente. Por consiguiente, se reduce la duración de la prueba. El inicio de la función OnTick() se hace en todos los puntos de control que se construyen a base de los precios OHLC de las barras minuteras.

La renuncia a generar los ticks intermedios adicionales entre los precios Open, High, Low y Close hace que aparece una determinación fuerte en el desarrollo del precio a partir del momento en el que ha sido determinado el precio Open. Esto da posibilidades para crear el "Grial de simulación" que muestra durante la simulación un bonito gráfico ascendiente. Puede encontrar un ejemplo de este Grial en Code Base - Grr-al.

© 2000-2025, MetaQuotes Ltd.



En la imagen podemos ver un gráfico muy atractivo de simulación de este EA. ¿Cómo ha salido así? Para una barra minutera se saben 4 precios. Y es sabido con seguridad que el precio Open va primero, y el último es el precio Close. Entre ellos hay precios High y Low, el orden de su aparición no se sabe, pero se sabe que el precio High es mayor o igual al precio Open (el precio Low es menor o igual al precio Open).

Basta con averiguar el momento de llegada del precio Open y luego analizar el siguiente tick con el fin de determinar qué es lo que tenemos delante, ¿High o Low? Si el precio es más bajo que el precio Open, eso significa que tenemos delante de nosotros el precio Low - entonces hacemos la compra en este tick. El siguiente tick va a corresponder al precio High en el que cerramos la compra y abrimos la venta. El siguiente tick es el último - es el precio Close, cerramos la venta en este tick.

Si después del precio ha llegado un tick con el precio que es más alto que el precio de apertura, entonces la secuencia de transacciones es inversa. Vamos a procesar en este modo tramposo la barra de un minuto y esperamos la siguiente. Mientras probamos este EA en los datos históricos, todo va perfectamente. Pero cuando lo hagamos en tiempo real, el cuento de hadas se desvanece -la línea del balance sigue siendo recta pero va hacia abajo. Para desvelar el truco, sólo hay que probar este EA en el modo "Todos los ticks".

> **Importante:** si los resultados de simulación del EA con el uso de los modos aproximados ("1 minute OHLC" y "Sólo precios de apertura") salen muy buenos, haga la prueba obligatoriamente en el modo "Todos los ticks".

Sólo precios de apertura

En este modo se generan los ticks sobre los precios OHLC del período (timeframe) seleccionado para la simulación. En este caso la función OnTick() del EA se inicia sólo al principio de la barra para el precio Open. Gracias a esta particularidad los niveles stop y las órdenes pendientes pueden iniciarse por el precio diferente al especificado (sobre todo durante la simulación en los períodos mayores). A cambio de eso, tenemos la posibilidad de llevar a cabo la simulación estimativa del EA de forma bastante rápida.

La excepción durante la simulación de ticks en el modo "Sólo precios de apertura" son los períodos W1 y MN1: para estos períodos los ticks se generan para los precios OHLC de cada día, y no para los precios OHLC de la semana y mes, respectivamente.

Por ejemplo, se hace la prueba del EA para EURUSD H1 en el modo "Sólo precios de apertura". En este caso, el número total de los ticks (puntos de control) no va a superar 4*número de barras de una hora que están dentro del intervalo de simulación. Pero la **llamada al manejador OnTick() se hace sólo en**

© 2000-2025, MetaQuotes Ltd.

el momento de la apertura de la **barra de una hora**. En los demás ticks ("invisibles" para el EA) se hacen las comprobaciones necesarias para la simulación correcta:

- cálculo de requerimientos de margen;
- accionamiento de Stop Loss y Take Profit;
- accionamiento de órdenes pendientes;
- eliminación de órdenes pendientes caducadas.

Si no hay posiciones abiertas u órdenes pendientes, entonces tampoco hay necesidad en estas comprobaciones para los ticks invisibles, y el aumento de la velocidad puede llegar a ser bastante importante. El modo "Sólo precios de apertura" conviene muy bien para probar las estrategias que realizan las transacciones sólo en la apertura de la barra, y no utilizan las órdenes pendientes ni tampoco las órdenes StopLoss,. TakeProfit. Para el tipo de estas estrategias se mantiene toda la precisión de simulación necesaria.

Como ejemplo de un EA para el que no importa el modo de simulación vamos a mostrar el EA Moving Average de la entrega estándar. La lógica de este EA está construida de tal manera que todas las decisiones se toman en la apertura de la barra y las transacciones se realizan enseguida, sin utilizar las órdenes pendientes. Vamos a arrancar la prueba del EA para EURUSD H1 en el intervalo desde 2010.01.09 hasta 2010.31.12, y compararemos los gráficos. En la imagen se muestran los gráficos del balance desde el informe del Probador para los tres modos.



Como puede ver, los gráficos de diferentes modos de simulación son absolutamente idénticos para el EA Moving Average desde la entrega estándar.

Existen ciertas limitaciones de la aplicación del modo "Sólo precios de apertura":

- No se puede utilizar el modo de trading "Retraso aleatorio";
- En el EA que se prueba no es posible acceder a los datos del inferior período que se utiliza para la simulación/optimización. Por ejemplo, si para la simulación/optimización se utiliza el período H1, Usted puede acceder a los datos del H2, H3, H4, etc., y no a los del M30, M20, M10, etc. Aparte de eso, los períodos mayores a los que se accede tienen que ser múltiplos del período de simulación.

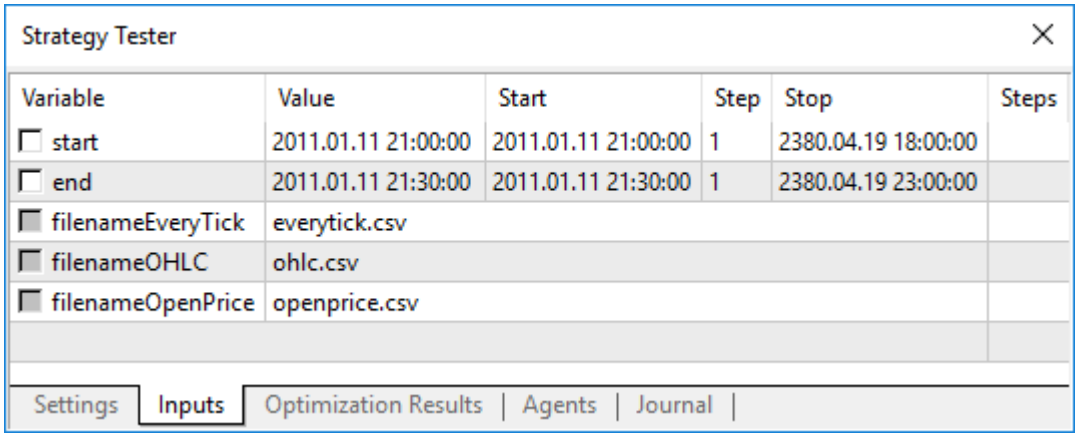
© 2000-2025, MetaQuotes Ltd.

Por ejemplo, durante la simulación en el período M20 no se puede acceder al período M30, pero se puede dirigirse al H1. Estas limitaciones están condicionadas a la imposibilidad de obtener los datos de los períodos inferiores y no múltiplos desde las barras que se generan durante la simulación/optimización.

- Las limitaciones del acceso a los datos de otros períodos se extienden también a otros símbolos cuyos datos utiliza el EA. No obstante, en este caso la limitación para cada símbolo depende del primer período al que se ha accedido durante la simulación/optimización. Por ejemplo, la simulación se lleva a cabo para el símbolo y período EURUSD H1, el EA se ha dirigido por primera vez al símbolo GBPUSD M20. En esta situación el EA puede utilizar en adelante los datos del EURUSD H1, H2, etc., así como los del GBPUSD M20, H1, H2, etc.

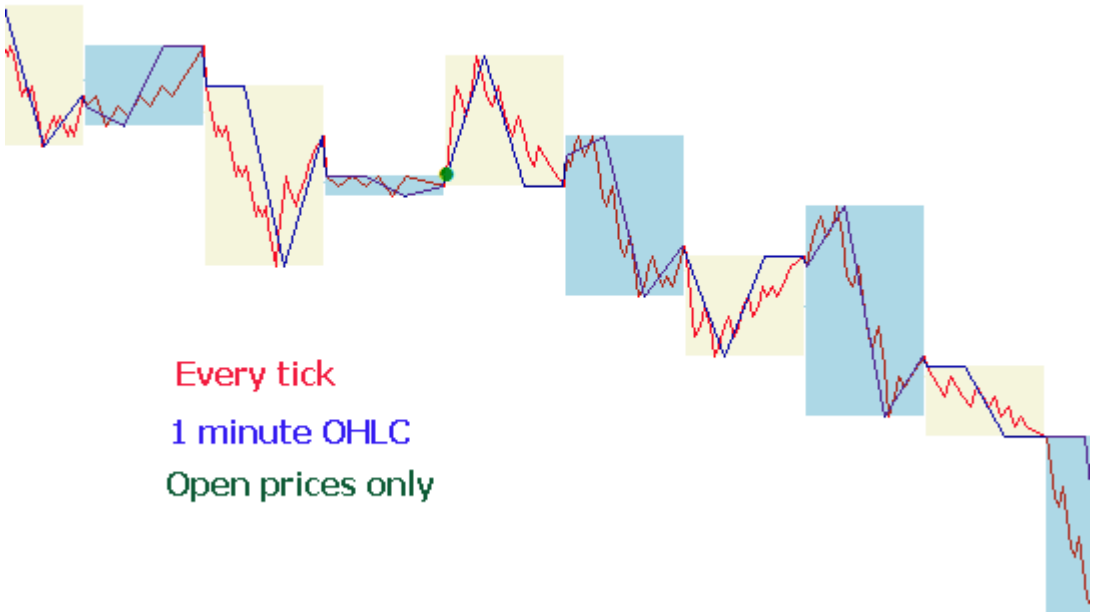
Importante: el modo "Sólo precios de apertura" es el más rápido por el tiempo que dura la simulación, pero no conviene para todas las estrategias de trading. Se debe seleccionar el modo de simulación necesario en función de las particularidades del sistema de trading que se utiliza.

Al final de esta sección que trata sobre los modos de modelación vamos a mostrar la comparación visual de diferentes modos de generación de ticks para EURUSD para dos barras de M15 en el intervalo 2011.01.11 21:00:00 - 2011.01.11 21:30:00. Los ticks ha sido escritos en archivos diferentes, utilizando el EA WriteTicksFromTester.mq5. La terminación de los nombres de estos archivos se establecen en los parámetros input filenamEveryTick, filenameOHLC y filenameOpenPrice.



Para obtener tres archivos con tres secuencias de ticks (para cada uno de los modos "Todos los ticks", "OHLC en barras minuteras" y "Sólo precios de apertura") el EA ha sido arrancado tres veces en los modos correspondientes en repasos únicos. Luego, utilizando el indicador TicksFromTester.mq5, los datos han sido proyectados al gráfico desde estos tres archivos. El código del indicador va adjunto al artículo.

© 2000-2025, MetaQuotes Ltd.



Por defecto, todas las operaciones con archivos en el lenguaje MQL5 se realizan dentro de los límites de una "sandbox de archivos", y durante la simulación el EA tiene disponible sólo su propia "sandbox de archivos". Para que el indicador y el EA puedan trabajar durante la simulación con los archivos de la misma carpeta, se utiliza la bandera FILE_COMMON. Ejemplo del código del EA:

```
//--- abrimos el archivo
   file=FileOpen(filename,FILE_WRITE|FILE_CSV|FILE_COMMON,";");
//--- comprobamos el éxito de la operación
   if(file==INVALID_HANDLE)
     {
      PrintFormat("No se ha podido abrir el archivo %s para la escritura. Código del e
      return;
     }
   else
     {
      //--- avisamos sobre el guardado en la carpeta compartida de todos los terminale
      PrintFormat("El archivo será guardado en la carpeta %s",TerminalInfoString(TERMI
     }
```

En el indicador para la lectura de datos también se ha utilizado la bandera FILE_COMMON, lo que ha permitido evitar el traspaso manual de los archivos necesarios de una carpeta a otra.

```
//--- abrimos el archivo
   int file=FileOpen(fname,FILE_READ|FILE_CSV|FILE_COMMON,";");
//--- comprobamos el éxito de la operación
   if(file==INVALID_HANDLE)
     {
      PrintFormat("No se ha podido abrir el archivo %s para la lectura. Código del err
      return;
     }
   else
     {
      //--- indicaremos la ubicación de la carpeta compartida de todos los terminales
      PrintFormat("El archivo será leído desde la carpeta %s",TerminalInfoString(TERMI
     }
```

© 2000-2025, MetaQuotes Ltd.

Modelación de spreads

La diferencia entre los precios Bid y Ask se llama spread. El spread no se modela durante la simulación, sino se coge de los datos históricos. If the spread is less than or equal to zero in the historical data, then the last known (at the moment of generation) spread of is used by testing agent.

En el Probador un spread siempre se considera como flotante. Es decir, SymbolInfoInteger(symbol, SYMBOL_SPREAD_FLOAT) siempre devuelve true.

Además, en los datos históricos se guardan los valores de los volúmenes de ticks y de los volúmenes comerciales. Para almacenar y obtener los datos, se utiliza una estructura especial MqlRates:

```
struct MqlRates
  {
   datetime time;         // fecha/hora de apertura de la barra
   double   open;         // precio de apertura Open
   double   high;         // precio máximo High
   double   low;          // precio mínimo Low
   double   close;        // precio de cierre Close
   long     tick_volume;  // volumen de ticks
   int      spread;       // spread
   long     real_volume;  // volumen de bolsa
  };
```

Uso de ticks reales durante la simulación

La simulación y optimización con ticks reales se aproxima al máximo a las condiciones reales. En lugar de ticks generados basados en los datos de minuto, se utilizan los ticks reales almacenados por el bróker. Dichos ticks provienen de la bolsa y los proveedores de liquidez.

Con objeto de garantizar una precisión mayor, también se utilizan las barras de minuto. En ellas se verifican y corrigen los datos de los ticks. De esta forma, también se evita la divergencia de los gráficos en el simulador y en el terminal del cliente.

El simulador comprueba la correspondencia entre los datos de los ticks y los parámetros de las barras de minuto, es decir: el tick no deberá exceder los precios High/Low de la barra; el tick que abre y cierra el minuto deberá coincidir con los precios Open/Close de la barra. También se compara el volumen. Si se detecta una disparidad, se descartarán los ticks correspondientes a esta barra de minutos. En lugar de ellos, se utilizarán los ticks generados (como en el modo "Cada tick").

Si en la historia del símbolo existe una barra de minuto sin datos de ticks para la misma, el simulador generará ticks en el modo "Todos los ticks". Esto permitirá dibujar correctamente el gráfico en el simulador, en el caso de que los datos de ticks del bróker estén incompletos.

Si en la hitoria del símbolo no existe una barra de minuto, pero sí hay datos de ticks apropiados para ese minuto, los datos pueden ser utilizados en el simulador. Por ejemplo, las barras de los símbolos de la bolsa se forman de acuerdo con los precios Last. Si desde el servidor llegan solo ticks con precios Bid/Ask pero sin precio Last, la barra no se formará. El simulador usará estos datos de ticks, ya que no contradicen las de minuto.

Los datos de ticks pueden diferenciarse de las barras de minuto por varios motivos. Por ejemplo, debido a desconexiones u otros fallos al transferirse al terminal de cliente los datos desde la fuente. Durante las simulaciones, los datos de minuto se consideran más fiables.

© 2000-2025, MetaQuotes Ltd.

Al simular con ticks reales, deberá tener en cuenta los siguientes aspectos:

- Al realizar la simulación, los datos de minuto del símbolo se sincronizarán junto con los datos de los ticks.
- Los ticks se almacenan en la caché del símbolo en el simulador de estrategias. El tamaño de la caché es inferior a los 128 000 ticks. Cuando llegan los nuevos ticks, los datos más antiguos de los mismos se expulsarán de la caché. Entre tanto, la función CopyTicks nos ayudará a obtener los ticks fuera de la caché (esto sucede en las simulaciones con ticks reales). En este caso, los datos se solicitarán a partir de la base de ticks del simulador, que se corresponde totalmente con la base análoga del terminal de cliente. En dicha base no se realiza ninguna corrección de las barras de minuto, por ello, los ticks en la misma pueden diferenciarse de los ticks en la caché.

Variables globales del Terminal de Cliente

Durante la simulación, las variables globales del Terminal de Cliente se emulan también, pero no están relacionadas de ninguna manera con auténticas variables globales del terminal que se puede ver en el terminal utilizando el botón F3. Eso quiere decir que todas las operaciones con las variables globales del terminal durante la simulación se realizan fuera del mismo (en el agente de pruebas).

Cálculo de indicadores durante la simulación

En el modo de tiempo real, los valores de los indicadores se calculan en cada tick.

En el simulador de estrategias, los indicadores se calculan solo al recurrir a los mismos, es decir, solo en el momento cuando se solicitan los valores de los búferes de indicador. La excepción son los indicadores personalizados con #property tester_everytick_calculate configurado; en dicho caso, el recálculo se realiza en cada tick.

En el modo de simulación visual, todos los indicadores se recalculan incondicionalmente al llegar un nuevo tick, para que así se muestren correctamente en el gráfico de simulación visual.

El cálculo del indicador en cada tick se realiza una vez, y todas las posteriores solicitudes de datos del indicador antes de que llegue un nuevo tick no provocan un nuevo cálculo. Por consiguiente, si el temporizador se ha habilitado en el EA usando la función EventSetTimer(), entonces antes de cada llamada del manejador OnTimer(), se solicitarán los datos del indicador desde el último tick. Si el indicador aún no se ha calculado en el último tick, se calcularán los valores del indicador. Si los datos ya han sido preparados, estos se ofrecerán sin realizar un nuevo recálculo.

Por ello, todos los cálculos de los indicadores se realizan de la forma más económica posible: si el indicador ya se ha calculado en un momento determinado, los datos del mismo se ofrecen tal cual, el indicador no se recalcula.

Carga del historial durante la simulación

El historial para el instrumento a probar se sincroniza y se descarga por el terminal desde el servidor comercial antes del inicio del proceso de simulación. En este caso, el terminal descarga desde el servidor comercial por primera vez todo el historial disponible para el instrumento a probar para luego ya no volver a este asunto. A continuación, se descargan sólo los datos nuevos.

El agente de pruebas recibe del Terminal de Cliente el historial para el instrumento a probar justamente después de que haya sido iniciado el proceso de simulación. Si durante el proceso de

© 2000-2025, MetaQuotes Ltd.

simulación se utilizan los datos de otros instrumentos (por ejemplo, si se trata de un EA de múltiples divisas), entonces en este caso el agente de pruebas solicita al Terminal de Cliente el historial necesario durante la primera invocación. Si los datos históricos se encuentran dentro del terminal, se pasan enseguida a los agentes de pruebas. Si no hay datos necesarios, el terminal los solicitará y descargará desde el servidor, y luego los pasará a los agentes de pruebas.

También se realiza el acceso a los instrumentos adicionales en el caso cuando se calcula el precio del tipo de cambio cruzado durante las operaciones de trading. Por ejemplo, durante la simulación de la estrategia sobre EURCHF con la moneda del depósito en dólares de los EE.UU. el agente de pruebas solicita al Terminal de Cliente el historial para EURUSD y USDCHF antes de procesar la primera operación de comercial, aunque la estrategia no supone la invocación directa a estos instrumentos financieros.

Antes de empezar a probar una estrategia de múltiples divisas, se recomienda descargar previamente todos los datos históricos necesarios al Terminal de Cliente. Esto permitirá evitar las demoras durante la simulación/optimización relacionadas con la descarga complementaria de datos que faltan. Por ejemplo, puede descargar el historial si abre los gráficos correspondientes y los desplaza hacia el inicio del historial. Puede encontrar un ejemplo de la descarga forzosa del historial al Terminal de Cliente en el apartado Organización de acceso a los datos de la documentación sobre MQL5.

Los agentes de pruebas en su lugar reciben el historial en forma comprimida desde terminal. Durante la simulación repetida el Probador ya no vuelve a descargar el historial desde el terminal, puesto que quedan los datos después del arranque anterior del Probador.

- El terminal descarga el historial desde el servidor comercial sólo una vez cuando el agente se dirige al terminal a por el historial para el símbolo a probar. El historial se descarga en forma comprimida con el fin de ahorrar el tráfico.
- Los ticks no se mandan por la red, sino se generan por los agentes de pruebas.

Simulación en múltiples divisas

El Probador permite llevar a cabo la simulación sobre el historial de estrategias que tradean utilizando varios instrumentos financieros. A estos EAs se les llaman condicionalmente de múltiples divisas, porque desde el principio en la plataformas anteriores la simulación se realizaba sólo para un instrumento financiero. Mientras que en el Probador del terminal MetaTrader 5 se puede simular el trading con todos los instrumentos disponibles.

El historial para los instrumentos utilizados se descarga por el Probador desde el **Terminal de Cliente** (¡no desde el servidor comercial!) de forma automática cuando se le llama al instrumento en cuestión por primera vez.

El agente de pruebas descarga sólo el historial que falta con pequeña reserva con el fin de asegurar los datos históricos necesarios para el cálculo de los indicadores en el momento de simulación. El volumen mínimo del historial a descargar desde el servidor comercial para los períodos D1 e inferiores es de un año. De esta manera, si se inicia la simulación en el intervalo 2010.11.01-2010.12.01 (simulación en el intervalo de un mes) con el período M15 (cada barra es igual a 15 minutos), entonces se le solicitará al terminal el historial para el instrumento durante el año 2010 entero. Para los períodos Weekly será solicitado el historial de 100 barras, lo que supone aproximadamente dos años (hay 52 semanas en el año). Para la simulación sobre el período mensual Monthly el agente solicitará el historial de 8 años (12 meses * 8 años = 96 meses).

© 2000-2025, MetaQuotes Ltd.

Si por alguna razón resulta imposible conseguir antes del inicio de la prueba el número de barras necesario para realizar la simulación, entonces la **fecha del inicio de la simulación será acercada automáticamente** hacia el presente para alcanzar esta reserva necesaria.

Durante la simulación se emula también la "Observación del Mercado" desde la cual se puede obtener la información sobre los instrumentos. Por defecto, al comienzo de la simulación la "Observación del Mercado" del Probador contiene sólo un símbolo, para el que ha sido iniciada la simulación. Todos los símbolos necesarios se conectan a la "Observación del Mercado" del Probador (¡no del Terminal!) de forma automática en cuanto se hace la llamada a ellos.

> Antes de empezar la simulación de un EA de múltiples divisas, hay que seleccionar los instrumentos necesarios para esta simulación en la "Observación del Mercado" del terminal y bajar los datos necesarios en la profundidad necesaria. Al llamar a un símbolo "ajeno" por primera vez, se ejecuta automáticamente la sincronización para este símbolo entre el agente de pruebas y el Terminal de Cliente. Un símbolo "ajeno" es aquél que se diferencia del símbolo sobre el que ha sido iniciada la simulación.

La llamada a los datos del símbolo ajeno se hace en las siguientes ocasiones:

- uso de las funciones de los indicadores técnicos y IndicatorCreate() sobre el par símbolo/período;
- llamada a la "Observación del Mercado" (Market Watch) para el símbolo ajeno:
  1. SeriesInfoInteger
  2. Bars
  3. SymbolSelect
  4. SymbolIsSynchronized
  5. SymbolInfoDouble
  6. SymbolInfoInteger
  7. SymbolInfoString
  8. SymbolInfoTick
  9. SymbolInfoSessionQuote
  10. SymbolInfoSessionTrade
  11. MarketBookAdd
  12. MarketBookGet
- llamada a las series temporales para el par símbolo/período utilizando las funciones:
  1. CopyBuffer
  2. CopyRates
  3. CopyTime
  4. CopyOpen
  5. CopyHigh
  6. CopyLow
  7. CopyClose
  8. CopyTickVolume
  9. CopyRealVolume

© 2000-2025, MetaQuotes Ltd.

10. CopySpread

En el momento cuando se hace la primera invocación a un símbolo ajeno, el proceso de simulación se detiene y se realiza la descarga adicional de datos históricos que faltan para el par símbolo/período desde el terminal al agente de pruebas. Al mismo tiempo se activa el proceso de generación de la secuencia de ticks para este símbolo.

Para cada instrumento se genera su propia secuencia de ticks de acuerdo con el modo de generación de ticks seleccionado. Aparte de eso, se puede solicitar el historial para los símbolos necesarios de forma explícita mediante la llamada a la función SymbolSelect() en el manejador OnInit(). En este caso el historial será descargado en el acto, antes que empiece a probar su EA.

De esta manera, para llevar a cabo la simulación de múltiples divisas en el Terminal de Cliente MetaTrader 5, no hace falta hacer ningunos esfuerzos adicionales. Bastará con abrir los gráficos de los instrumentos correspondientes en el Terminal de Cliente. El historial de los símbolos necesarios se descargará automáticamente desde el servidor comercial con la condición si dispone de estos datos.

Modelación de la hora en el Probador

Durante la simulación la hora local TimeLocal() siempre es iguala a la hora del servidor TimeTradeServer(). En su lugar, la hora del servidor siempre es iguala la hora que corresponde a la hora GMT - TimeGMT(). Así, durante la simulación todas estas funciones muestran la misma hora.

La falta de diferencia entre GMT, la hora local y de servidor en el Probador está hecha a propósito debido a que la conexión con el servidor no siempre puede ser permanente. Mientras que los resultados de la simulación tienen que ser iguales, independientemente de que si hay conexión o no. La información sobre la hora de servidor no se guarda de forma local, sino se coge en el servidor.

Objetos gráficos durante la simulación

La construcción de los objetos gráficos no se hace durante la simulación/optimización. De esta manera, el EA obtiene los valores cero cuando se dirige a las propiedades del objeto creado durante la simulación/optimización.

> Esta limitación no concierne a la simulación en modo visual.

Función OnTimer() en el Probador

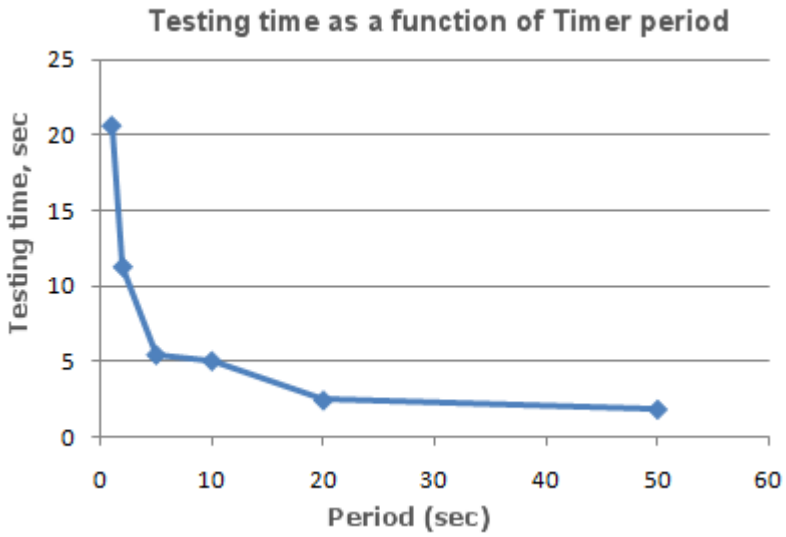
En MQL5 se puede procesar los eventos del temporizador. La llamada al manejador OnTimer() se realiza independientemente del modo de simulación. Eso significa que si la simulación ha sido iniciada en el modo "Sólo los precios de apertura" sobre el período H4 y dentro del EA está instalado un temporizador con la llamada cada segundo, entonces durante la apertura de cada barra H4 el manejador OnTick() será llamado una vez, y 14400 veces (3600 segundos * 4 horas) durante la barra será llamado el manejador OnTimer(). En cuánto va a aumentarse el tiempo de simulación depende de la lógica del EA.

Hemos escrito un simple EA sin operaciones comerciales para comprobar la dependencia del tiempo de simulación de la periodicidad del temporizador.

© 2000-2025, MetaQuotes Ltd.

```
//--- input parameters
input int      timer=1;              // valor del temporizador, segundos
input bool     timer_switch_on=true; // temporizador activado
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- iniciamos el temporizador si timer_switch_on==true
   if(timer_switch_on)
     {
      EventSetTimer(timer);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- detenemos el temporizador
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
// no hacemos nada, el cuerpo del manejador está vacío
  }
//+------------------------------------------------------------------+
```

Se ha medido el tiempo de simulación con diferentes valores del parámetro timer (periodicidad del evento Timer). A base de los datos obtenidos se ha construido el gráfico de dependencia del tiempo de simulación T del valor de periodicidad Period.



Aquí se ve muy bien, cuanto más bajo sea el parámetro timer en el momento de inicialización del temporizador por la función EventSetTimer(timer), es menor el período (Period) entre las llamadas al

© 2000-2025, MetaQuotes Ltd.

manejador OnTimer(), y es mayor el tiempo de simulación T durante las mismas condiciones restantes.

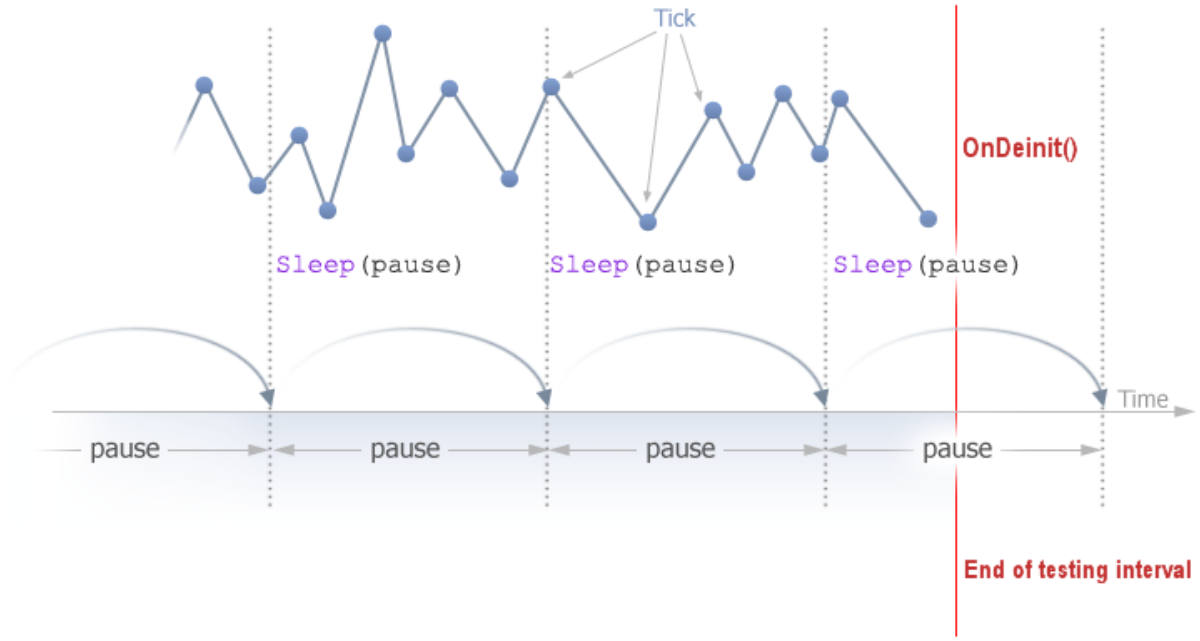
Función Sleep() en el Probador

La función Sleep() permite parar temporalmente la ejecución del programa mql5 en el EA o script durante el trabajo en el gráfico. Esto puede ser útil cuando se solicitan algunos datos que en el momento de la solicitud aún no están listos y hace falta esperar hasta que estén disponibles. Puede encontrar un ejemplo detallado del uso de la función Sleep() en el apartado Organización de acceso a los datos.

Pero en el Probador las llamadas a la función Sleep() no retrasan el proceso de simulación. Cuando se llama a la función Sleep(), "se reproducen" los ticks generados dentro del margen del retraso especificado, como resultado de lo cual pueden accionarse las órdenes pendientes, stops, etc. Después de la llamada a la función Sleep(), el tiempo modelado en el Probador se aumenta al intervalo especificado en el parámetro de la función Sleep.

Si como resultado de la ejecución de la función Sleep() el tiempo actual en el Probador ha salido fuera del período de simulación, verá el mensaje del error "ciclo infinito en Sleep". En caso de este error, los resultados de la simulación no se omiten. Todos los cálculos se realizan íntegramente (número de transacciones, reducciones, etc.).

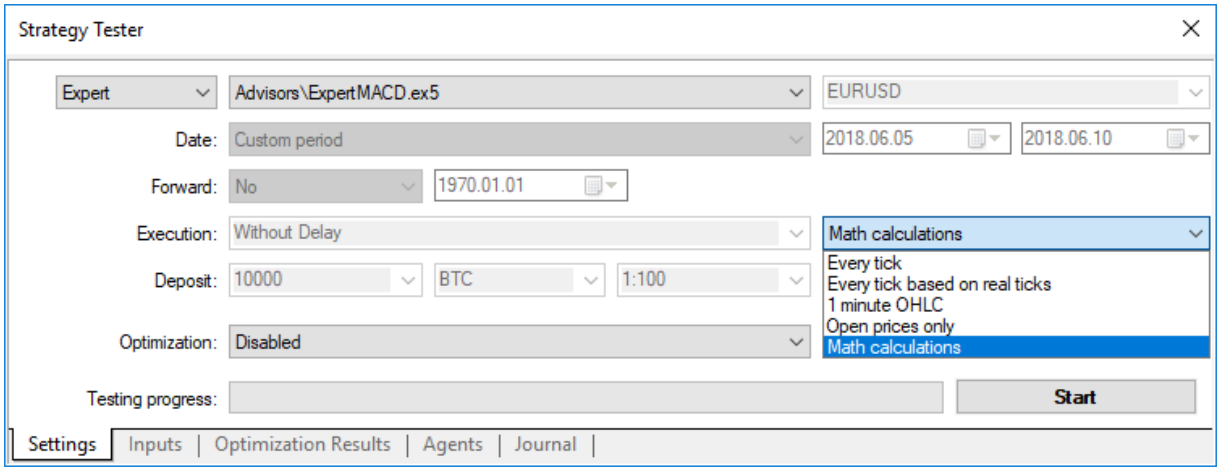
La función Sleep() no va a funcionar en OnDeinit(), ya que tras su llamada el tiempo de simulación sí o sí saldrá fuera del intervalo de simulación.



Uso del Probador para las tareas de optimización en los cálculos matemáticos

En el terminal MetaTrader 5 se puede utilizar el Probador no sólo para probar las estrategias comerciales, sino también para los cálculos matemáticos. Para eso hay que seleccionar el modo correspondiente en las opciones:

© 2000-2025, MetaQuotes Ltd.



Al elegir el modo "Cálculos matemáticos", será realizado un repaso en "vacío" del agente de simulación. El repaso en "vacío" significa que no se realizará la generación de los ticks ni tampoco va a cargarse el historial. Durante este repaso sólo serán llamadas las funciones OnInit(), OnTester() y OnDeinit().

Si la fecha de finalización de la prueba es menor o igual a la fecha de su inicio, esto también va a significar la simulación en el modo "Cálculos matemáticos".

Durante el uso del Probador para la solución de tareas matemáticas, la descarga del historial y la generación de los ticks no se hace.

Una tarea matemática muy típica a resolver en el Probador del MetaTrader 5 es la búsqueda del extremo de la función de muchas variables. Para su solución hace falta:

- Colocar el bloque de cálculos del valor de la función de muchas variables en OnTester(), y devolver el valor calculado a través de return(valor_de_la_función);
- Pasar los parámetros de la función al área global del programa en forma de las variables input;

Compilamos el EA, abrimos la ventana "Probador". En la pestaña "Parámetros de entrada" marcamos los parámetros de entrada necesarios y establecemos para ellos los límites en el espacio de los valores y el paso para el repaso.

Seleccionamos el tipo de optimización: "Lenta" (repaso completo de parámetros) o "Rápida" (algoritmo genético). Es mejor seleccionar la optimización rápida para la simple búsqueda del extremo de la función. Pero si hace falta calcular los valores en todo el espacio de las variables, mejor conviene la optimización lenta.

Seleccionamos el modo "Cálculos matemáticos" e iniciamos el proceso de optimización haciendo clic en el botón "Empezar". Hay que recordar que durante la optimización siempre se busca el máximo local del valor de la función OnTester. Para la búsqueda del mínimo local se puede devolver de la función OnTester el valor inverso al valor calculado de la función:

```
return(1/valor_de_la_función);
```

En este caso Usted mismo debe comprobar que el valor_de_la_función no sea igual a cero, ya que de lo contrario se puede recibir un error crítico de división por cero. Hay otra opción que es más conveniente y que no altera los resultados de la optimización, ha sido ofrecido por los lectores del artículo:

```
return(-valor_de_la_función);
```

© 2000-2025, MetaQuotes Ltd.

Aquí no hace falta comprobar si el valor_de_la_función es igual a cero, y la misma superficie de los resultados de la optimización en representación 3D tiene la misma forma pero reflejada de espejo respecto a la inicial.

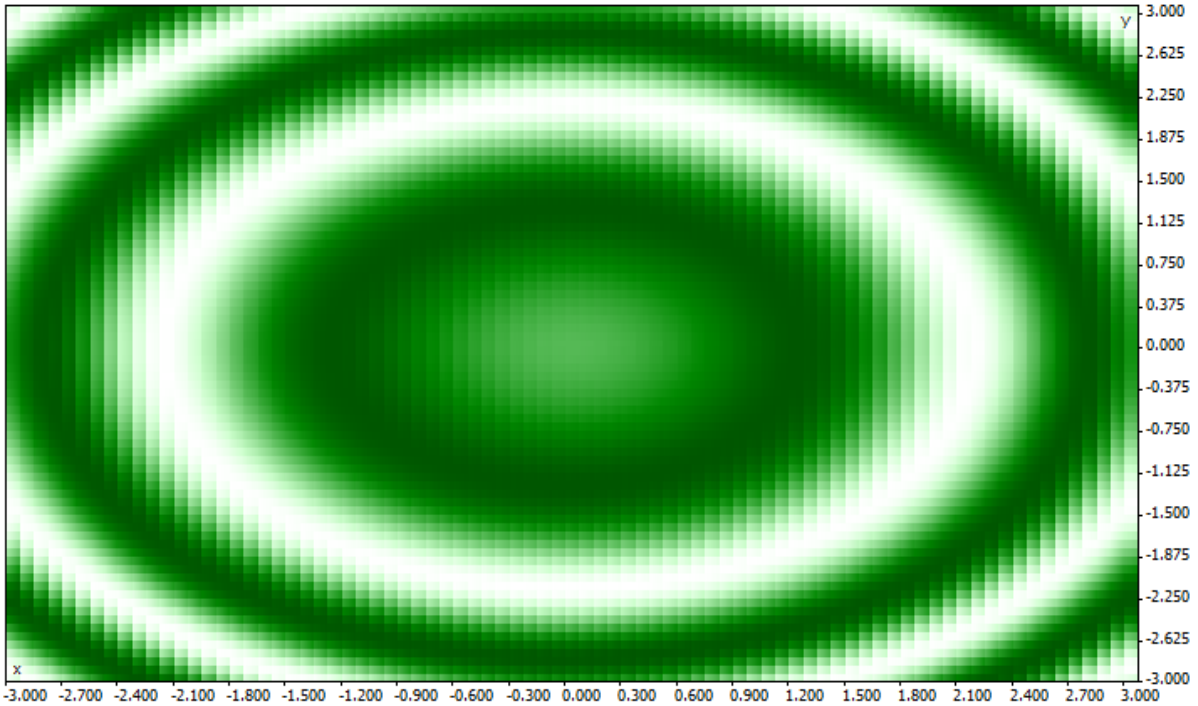
Como ejemplo vamos a coger la función sink():

$$sink(x,y) = \sin(x^2 + y^2)$$

El código del EA para la búsqueda del extremo de esta función vamos a colocar en OnTester():

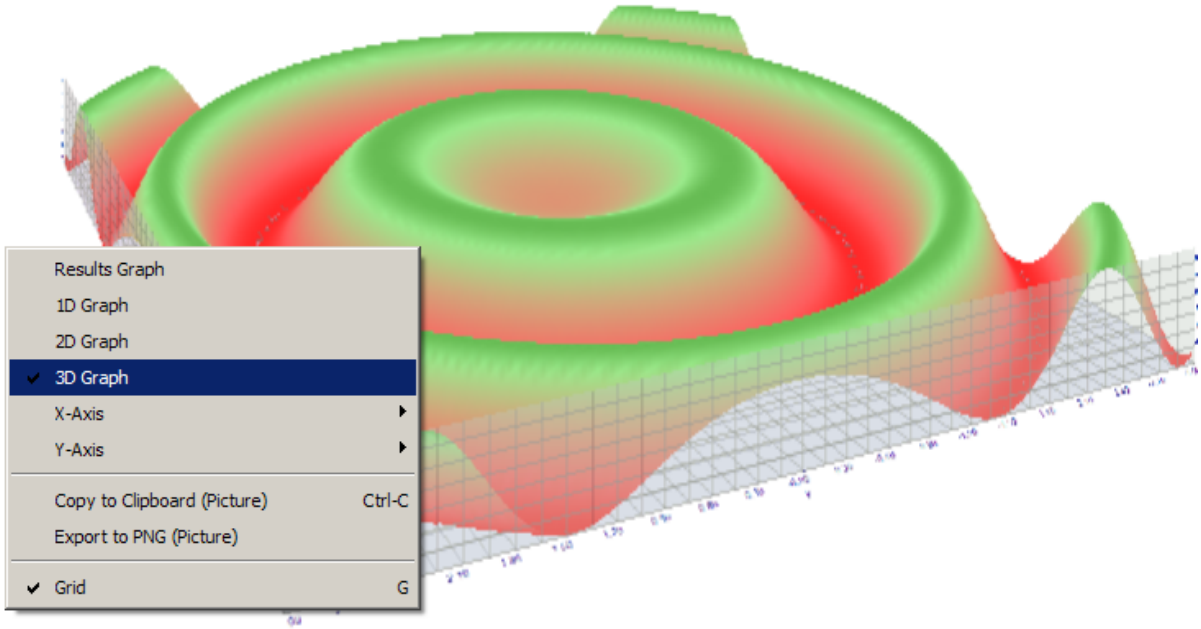
```
//+------------------------------------------------------------------+
//|                                                         Sink.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- input parameters
input double   x=-3.0; // start=-3, step=0.05, stop=3
input double   y=-3.0; // start=-3, step=0.05, stop=3
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//---
   double sink=MathSin(x*x+y*y);
//---
   return(sink);
  }
//+------------------------------------------------------------------+
```

Hagamos la optimización y mostremos los resultados de la optimización como un gráfico 2D.



© 2000-2025, MetaQuotes Ltd.

Cuanto mejor sea el valor para el par de parámetros establecido (x, y), más denso será el color. Tal como se esperaba partiendo de la vista de la fórmula de la función sink(), sus valores forman unos círculos concéntricos con el centro en el punto (0,0). Para la función sink() no existe un extremos absoluto. Esto se ve muy bien cuando vemos los resultados de optimización en el modo 3D:



Sincronización de las barras durante la simulación en el modo "Sólo precios de apertura"

El Probador en el terminal MetaTrader 5 permite probar también así llamados Asesores Expertos "de múltiples divisas". Un EA de múltiples divisas es un Asesor Experto que opera con dos o más símbolos.

La simulación de las estrategias que tradean con varios instrumentos impone al Probador unos requerimientos técnicos adicionales:

- generación de ticks para estos instrumentos;
- cálculo de valores de los indicadores para estos instrumentos;
- cálculo de requerimientos del margen para estos instrumentos;
- sincronización de secuencias de ticks generadas para todos los instrumentos con los que se tradea.

El Probador genera y reproduce una secuencia de ticks para cada instrumento en función del modo de trading seleccionado. En este caso la nueva barraen cada instrumento se abre independientemente de cómo se ha abierto la barra en otro instrumento. Eso significa que durante la simulación de un EA de múltiples divisas puede surgir la situación (suele pasar con bastante frecuencia) cuando en un instrumento la barra ya se ha abierto y en el otro todavía no. De esta manera, durante la simulación pasa lo mismo que pasa en la vida real.

Esta auténtica modelación del desarrollo del historial en el Probador no causa preguntas hasta que utilizamos los modos de simulación "Todos los ticks" y "1 minute OHLC". Durante el uso de estos modos, dentro de los límites de una vela se genera una cantidad de ticks suficiente para esperar el momento de sincronización de las barras de diferentes símbolos. ¿Pero cómo vamos a probar las estrategias de múltiples divisas en el modo "Sólo precios de apertura" si se requiere la sincronización

© 2000-2025, MetaQuotes Ltd.

obligatoria de las barras para los instrumentos en los que tradeamos? Pues, en este modo el EA es llamado sólo en el tick que correspondiente a la hora de apretura de la barra.

Lo explicaremos en un ejemplo: si probamos nuestro EA sobre el símbolo EURUSD, y en EURUSD ha sido abierta una nueva vela, será muy fácil enterarnos de eso. Es que durante la simulación en el modo "Sólo precios de apertura", el evento NewTick corresponde al momento de apertura de la barra en el período de la prueba. Pero no existe garantía alguna de que la nueva vela se ha abierto para el símbolo GBPUSD que se utiliza en el EA.

En condiciones normales bastará con finalizar el trabajo de la función OnTick() y comprobar la aparición de la nueva barra para GBPUSD durante el siguiente tick. Pero durante la simulación en el modo "Sólo precios de apertura" no habrá ningún otro tick, y a lo mejor puede formarse la impresión que este modo no vale para probar los EAs de múltiples divisas. Pero eso no es así. No olvide que el Probador en MetaTrader 5 se comporta igual como en la vida real. Se puede esperar el momento cuando para el otro símbolo se abrirá una nueva barra mediante la función Sleep()!

Este es código del EA Synchronize_Bars_Use_Sleep.mq5 que muestra el ejemplo de sincronización de las barras durante la simulación en el modo "Sólo precios de apertura":

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                  Synchronize_Bars_Use_Sleep.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- input parameters
input string   other_symbol="USDJPY";
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- confrontamos el símbolo actual
   if(_Symbol==other_symbol)
     {
      PrintFormat("¡Hace falta especificar otro símbolo o iniciar la simulación sobre
      //--- finalizamos la prueba forzosamente
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- variable estática para guardar la hora de apertura de la última barra
   static datetime last_bar_time=0;
//---  indicio de que la hora de apertura de la última barra de diferentes símbolos es
   static bool synchonized=false;
//--- si la variable estática aún no está inicializada
   if(last_bar_time==0)
     {
      //--- es la primera llamada, apuntaos la hora de apertura y salimos
      last_bar_time=(datetime)SeriesInfoInteger(_Symbol,Period(),SERIES_LASTBAR_DATE);
      PrintFormat("Hemos inicializado la variable last_bar_time con el valor %s",TimeT
     }
//--- obtenemos la hora de apertura de la última barra para nuestro símbolo
   datetime curr_time=(datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DAT
//--- si la hora de apertura de la barra actual no coincide con la que se guarda en la
   if(curr_time!=last_bar_time)
     {
      //--- recordamos la hora de apertura de la nueva barra en la variable estática
      last_bar_time=curr_time;
      //--- la sincronización ha sido violada, mostramos la bandera false
      synchonized=false;
      //--- mostramos el mensaje sobre este evento
      PrintFormat("Para el símbolo %s se ha abierto nueva barra a las %s",_Symbol,Time
     }
//--- aquí vamos a guardar la hora de apertura de la barras para el símbolo ajeno
   datetime other_time;
//--- ciclo, hasta que la hora de apertura de la última barra para el otro símbolo co
   while(!(curr_time==(other_time=(datetime)SeriesInfoInteger(other_symbol,Period(),SE
     {
      PrintFormat("Esperaremos 5 segundos..");
      //--- esperaremos 5 segundos y volveremos a solicitar SeriesInfoInteger(other_sy
      Sleep(5000);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- la hora de apertura de la barra ahora es igual para los dos símbolos
   synchonized=true;
   PrintFormat("La hora de apertura de la última barra para nuestro símbolo %s: %s",_S
   PrintFormat("La hora de apertura de la última barra para el símbolo %s: %s",other_s
//--- TimeCurrent() no vale, utilizamos TimeTradeServer() para
   Print("Las barras han sido sincronizadas a las ",TimeToString(TimeTradeServer(),TIM
  }
//+------------------------------------------------------------------+
```

Fíjense en la última línea del EA que nos muestra la hora actual a la que ha sido determinado el hecho de sincronización:

```
   Print("Las barras han sido sincronizadas a las ",TimeToString(TimeTradeServer(),TIM
```

Para mostrar la hora actual hemos utilizado la función TimeTradeServer(), en vez de la TimeCurrent(). Es que la función TimeCurrent() devuelve la hora del último tick que no se ha cambiado de ninguna manera tras el uso de Sleep(). Inicie el EA en el modo "Sólo precios de apertura" y verá los mensajes sobre la sincronización de las barras.

Core 1	2010.12.01 20:00:05	The bars are synchronized at 2010.12.01 20:00:05
Core 1	2010.12.01 20:00:05	Open bar time of the chart symbol USDJPY: 2010.12.01 20:00
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 20:00:00	Waiting 5 seconds..
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 16:00:05	The bars are synchronized at 2010.12.01 16:00:05

Utilice la función TimeTradeServer() en lugar de la TimeCurrent() si necesita obtener la hora de servidor actual y no la hora de llegada del último tick.

Hay otra forma de sincronizar las barras - utilizando el temporizador. El ejemplo de tal EA Synchronize_Bars_Use_OnTimer.mq5 se adjunta al artículo.

Función IndicatorRelease() en el Probador

Después de la finalización de la simulación, se abre automáticamente el gráfico del instrumento en el cual se muestran las transacciones realizadas y los indicadores que se han utilizado en el EA. Esto ayuda comprobar de forma visual los momentos de entrada y salida, así como compararlos con los valores de los indicadores.

> **Importante:** los indicadores mostrados en el gráfico abierto automáticamente tras finalizarse la simulación se calculan de nuevo ya después de completarse la prueba. Incluso si estos indicadores han sido utilizados en el EA a probar.

Pero en algunas ocasiones el programador puede necesitar ocultar la información sobre los indicadores utilizados en el algoritmo de trading. Por ejemplo, el código del EA se alquila o se vende como archivo ejecutable sin proporcionar el código fuente. Para este propósito valdrá la función IndicatorRelease().

Si en la carpeta /profiles/templates del Terminal de Cliente hay una plantilla que se llama tester.tpl, precisamente esta plantilla será aplicada al gráfico que se abre. Si no hay esta plantilla, se aplica la plantilla predeterminada (default.tpl).

© 2000-2025, MetaQuotes Ltd.

Desde el principio la función IndicatorRelease() está destinada para liberar la parte de cálculo del indicador, en caso de que ya no es necesario. Esto permite ahorrar la memoria y los recursos de la CPU, porque cada tick activa el cálculo del indicador. Su segundo cometido consiste en prohibir la visualización del indicador en el gráfico de simulación tras finalizarse el repaso único.

Para prohibir la visualización del indicador en el gráfico después de la simulación, invoque la función IndicatorRelease() con el handle del indicador en el manejador OnDeinit(). La función OnDeinit() siempre se invoca después de la finalización y antes de visualización del gráfico de simulación.

```
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   bool hidden=IndicatorRelease(handle_ind);
   if(hidden) Print("IndicatorRelease() ejecutada con éxito");
   else Print("IndicatorRelease() ha devuelto false. Código del error ",GetLastError()
  }
```

Para prohibir la visualización del indicador en el gráfico después de la simulación, utilice la función IndicatorRelease() en el manejador OnDeinit().

Procesamiento de eventos en el Probador

La presencia del manejador OnTick() en el EA no es obligatoria para que se pueda pobarlo sobre los datos históricos en el Probador del terminal MetaTrader 5. Será suficiente que en el EA haya por lo menos una función-manejador listadas más abajo:

- OnTick() - manejador del evento de la llegada de un nuevo tick;
- OnTrade() - manejador de un evento comercial;
- OnTimer() - manejador del evento de la llegada de una señal de temporizador;
- OnChartEvent() - manejador de los eventos del usuario.

Durante la simulación, en el Probador se puede procesar los eventos de usuario utilizando la función OnChartEvent(), pero en los indicadores esta función no se invoca en el Probador. Incluso si el indicador dispone del manejador OnChartEvent() y este indicador se utiliza en el EA, este indicador no va a recibir ningunos eventos personalizados.

Durante la simulación el indicador puede generar los eventos personalizados utilizando la función EventChartCustom(), y el EA puede procesar este evento en OnChartEvent().

Aparte de los eventos arriba mencionados en el Probador de Estrategias se generan los eventos especiales relacionados con el proceso de simulación y optimización:

- Tester - este evento se genera cuando se finaliza la simulación del EA a base de los datos históricos. El procesamiento del evento Tester se hace por la función OnTester(). Se puede utilizar esta función sólo en los EAs durante la simulación, y en primer lugar está destinada para calcular un valor que se utiliza como criterio Custom max durante la optimización genética de los parámetros de entrada.
- TesterInit - este evento se genera cuando se inicia el proceso de optimización en el Probador de Estrategias antes del primer repaso. El procesamiento del evento TesterInit se realiza por la función OnTesterInit(). El EA que dispone de este manejador se carga automáticamente, al iniciarse la optimización, en un gráfico nuevo del terminal con el símbolo y período especificados en el

© 2000-2025, MetaQuotes Ltd.

Probador, y recibe el evento TesterInit. La función está destinada para inicializar el EA antes del inicio de la optimización para el posterior procesamiento de los resultados de la optimización.

- TesterPass - este evento se genera cuando llega un nuevo frame de datos. El procesamiento del evento TesterPass se realiza por la función OnTesterPass(). El EA con este manejador se carga automáticamente en un gráfico nuevo del terminal con el símbolo/período especificados para la simulación, y recibe durante la optimización el evento TesterPass cuando llegue un frame. La función está destinada para el procesamiento dinámico de los resultados de la optimización directamente "al vuelo", sin esperar su finalización. La agregación de los frames se realiza por la función FrameAdd(), que puede ser invocada cuando se finaliza el repaso único en el manejador OnTester().
- TesterDeinit - este evento se genera cuando se termina el proceso de optimización del EA en el Probador de Estrategias. El procesamiento del evento TesterDeinit se realiza por la función OnTesterDeinit(). El EA con este manejador se carga automáticamente en el gráfico al iniciarse la optimización y recibe el evento TesterDeinit tras su finalización. Esta función está destinada para el procesamiento final de todos los resultados de la optimización.

Agentes de pruebas

En el Terminal de Cliente MetaTrader 5 la simulación se realiza utilizando los agentes de pruebas. Los agentes locales se crean y se conectan de forma automática. Por defecto, el número de los agentes locales corresponde al número de núcleos que tiene el ordenador.

Cada agente de pruebas dispone de su propia copia de variables globales que no está relacionada de ninguna manera con el Terminal de Cliente. El mismo terminal desempeña el papel del operador que reparte las tareas para los agentes locales y remotos. Después de ejecutar la tarea de turno relacionada con la simulación de un EA con parámetros establecidos, el agente devuelve el resultado al terminal. Durante la prueba única se utiliza sólo un agente.

El agente guarda el historial que recibe del terminal en las carpetas separadas que llevan el nombre del instrumento. Es decir, el historial para EURUSD se guarda en la carpeta con el nombre EURUSD. Aparte de eso el historial de los instrumentos se separa según las fuentes. La estructura de almacenamiento del historial es la siguiente:

```
carpeta_del_probador\Agent-IPaddress-Port\bases\nombre_de_la_fuente\history\nombre_del
```

Por ejemplo, el historial para EURUSD del servidor MetaQuotes-Demo se puede guardar en la carpeta_del_probador\Agent-127.0.0.1-3000\bases\MetaQuotes-Demo\EURUSD.

Después de haberse finalizado el proceso de simulación, el agente local se encuentra durante cinco minutos en el modo de espera de la siguiente tarea para no perder tiempo con el arranque en caso de las siguientes llamadas. Y sólo transcurrido este plazo de espera, el agente local finaliza su trabajo y se descarga de la memoria del ordenador.

En caso de la finalización anticipada de la simulación por parte del usuario (el botón "Cancelar"), así como en caso del cierre del terminal, todos los agentes locales finalizan su trabajo y se descargan de la memoria del ordenador.

Intercambio de datos entre el Terminal y el Agente

Cuando se inicia el proceso de simulación, el terminal se prepara para enviar al agente unos bloques de parámetros:

© 2000-2025, MetaQuotes Ltd.

- Parámetros de entrada de simulación (modo de modelación, intervalo de simulación, instrumento, criterio de optimización, etc.)
- Lista de instrumentos seleccionados en "Observación del Mercado"
- Especificación del instrumento a probar (tamaño del contrato, desviaciones permitidas del mercado para colocar StopLoss y Takeprofit, etc.)
- EA a probar y los valores de sus parámetros de entrada
- Información sobre los archivos adicionales (bibliotecas, indicadores, archivos de datos - #property tester_...)

tester_indicator	string	Nombre del indicador personalizado en el formato "nombre_del_indicador.ex5". Los indicadores necesarios para la simulación se determinan automáticamente desde la llamada de la función iCustom(), si el parámetro correspondiente ha sido establecido con una constante literal. Para los demás casos (el uso de la función IndicatorCreate() o el uso de una cadena no constante en el parámetro que establece el nombre del indicador) hace falta esta propiedad
tester_file	string	Nombre del archivo para el Probador con extensión, encerrado entre dobles comillas (como constante literal). El archivo especificado será pasado al Probador. Siempre hay que especificar los archivos de entrada para la simulación, en caso de que haya necesidad de ellos
tester_library	string	Nombre de la biblioteca con extensión encerrado entre dobles comillas. Una biblioteca puede tener la extensión dll o ex5. Las bibliotecas necesarias para la simulación se determinan automáticamente. Sin embargo, si alguna biblioteca se utiliza por un indicador personalizado, esta propiedad es necesaria

Para cada bloque de parámetros se crea una huella digital en forma del hash MD5 que se envía al agente. El hash MD5 es único para cada conjunto de datos, siendo su volumen muchas veces menor que el volumen de información a base de la cual éste ha sido calculado.

El agente recibe los hashs de los bloques y los compara con los que ya tiene almacenados. Si el agente no dispone de la huella de este bloque de parámetros, o el hash recibido se diferencia del existente, el agente solicita el bloque de parámetros en sí. De esta manera, se reduce el volumen de tráfico entre el terminal y el agente.

Después de realizar la simulación, el agente devuelve al terminal todos los resultados de la prueba que se muestran en las pestañas "Resultados de simulación" y "Resultados de optimización": beneficio obtenido, número de transacciones, ratio de Sharpe, resultado de la función OnTester(), etc.

Durante la optimización, el terminal reparte entre los agentes las tareas para realizar la prueba utilizando pequeños paquetes de datos. Cada paquete contiene unas cuantas tareas (cada tarea supone una prueba única con el conjunto de parámetros de entrada). Esto reduce el tiempo de intercambio entre el terminal y el agente.

Los agentes nunca guardan en el disco duro los archivos EX5 recibidos del terminal (EA, indicadores, bibliotecas, etc.) por motivos de seguridad. Se hace para que no se pueda utilizar los datos recibidos

© 2000-2025, MetaQuotes Ltd.

en el ordenador con el agente instalado. Todos los demás datos, incluyendo DLL, se guardan en la zona protegida (sandbox). En los agentes remotos no se puede probar los EAs con el uso de las DLL.

El terminal deposita los resultados de simulación en una caché de resultados especial (caché resultante) para un acceso rápido a ellos cuando surja esta necesidad. Para cada conjunto de parámetros el terminal busca en la caché resultante los resultados ya listos de los arranques anteriores con el fin de evitar los arranques repetitivos. Si el resultado con este conjunto de parámetros no ha sido encontrado, el agente recibe la orden para empezar la prueba.

Todo el tráfico entre el terminal y el agente se codifica.

> Los ticks no se mandan por la red, sino se generan por los agentes de pruebas.

Uso de la carpeta compartida de todos los Terminales de Cliente

Todos los agentes de pruebas están aislados uno del otro y del Terminal de Cliente también: cada agente tiene su propia carpeta donde se guardan todos los logs del agente. Además de eso, durante la simulación todas las operaciones con los archivos se hacen en la carpeta **nombre_del_agente/MQL5/Files**. No obstante, se puede organizar la interacción entre los agentes locales y el terminal a través de la carpeta compartida de todos los terminales de cliente si durante la apertura del archivo indicamos la bandera FILE_COMMON:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- carpeta compartida de todos los Terminales de Cliente
   common_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- mostraremos el nombre de esta carpeta
   PrintFormat("Abrimos el archivo en la carpeta compartida de todos los Terminales de
//--- abrimos el archivo en la carpeta compartida (tenemos la bandera FILE_COMMON)
   handle=FileOpen(filename,FILE_WRITE|FILE_READ|FILE_COMMON);
  ... siguientes acciones
//---
   return(INIT_SUCCEEDED);
  }
```

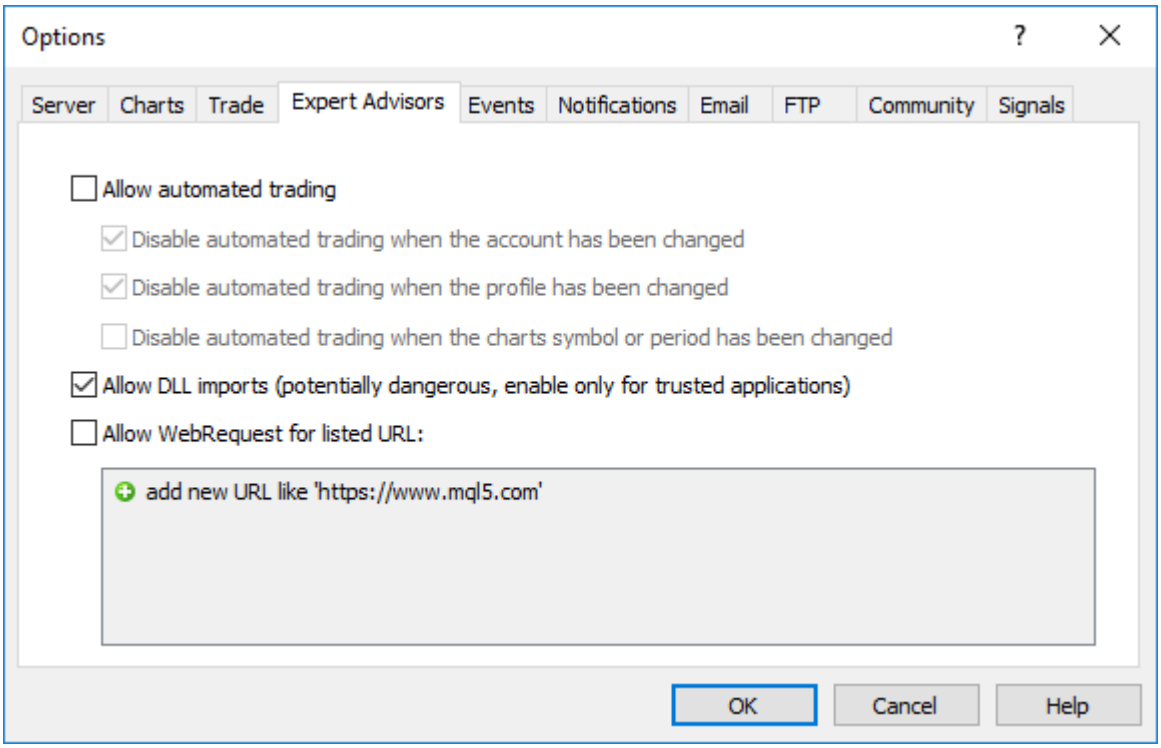
Uso de la DLL

Para acelerar el proceso de la simulación, se puede utilizar no sólo los agentes locales, sino también los agentes remotos. Pero hay ciertas limitaciones para los agentes remotos. En primer lugar, los agentes remotos no muestran en sus logs los resultados de ejecución de la función Print(), los mensajes sobre la apertura y el cierre de posiciones. En el log se muestra el mínimo de información para que los EAs escritos de forma incorrecta no sobrellenen con sus mensajes el disco duro del ordenador en el que trabaja el agente remoto.

La segunda limitación consiste en prohibición del uso de DLL durante la simulación de los EAs. Las llamadas a las bibliotecas DLL están completamente prohibidas en los agentes remotos por motivos de seguridad. Para los agentes locales las llamadas dll dentro de los EAs que se prueban están permitidas

© 2000-2025, MetaQuotes Ltd.

sólo en el caso si el permiso correspondiente ha sido concedido por medio de la opción "Permitir importación de DLL".



Importante: si utiliza los EAs (scripts, indicadores) recibidos desde fuera que exigen que les permita las llamadas a DLL, Usted debe comprender todo el riesgo que toma sobre sí en caso de permitir el uso de esta opción en los ajustes del terminal. Y eso no depende de forma del uso del EA -para la simulación o para su arranque en el gráfico.

© 2000-2025, MetaQuotes Ltd.

Variables predefinidas

Para cada programa MQL5 en ejecución se soporta una serie de variables predefinidas. Éstas reflejan el estado de gráfico de precios corriente en el momento cuando el programa (Experto, script o indicador personalizado) se inicie.

El terminal de usuario establece los valores para las variables predefinidas antes de que se inicie el programa mql5. Las variables predefinidas son constantes y no pueden ser cambiadas desde el programa mql5, salvo la variable _LastError, la cual puede ser anulada por la función ResetLastError.

Variable	Valor
_AppliedTo	La variable _AppliedTo permite conocer en el indicador el tipo de datos sobre los que se calcula
_Digits	Número de dígitos después de la coma decimal
_Point	Tamaño del punto del instrumento corriente en divisa de cotización
_LastError	Valor del último error
_Period	Valor del período de gráfico corriente
_RandomSeed	Estado actual del generador de números pseudoaleatorios
_StopFlag	Bandera de detención del programa
_Symbol	Nombre del símbolo del gráfico corriente
_UninitReason	Código de la causa de deinicialización
_IsX64	La variable _IsX64 permite saber en qué terminal se ha iniciado un programa MQL5

Las bibliotecas utilizan las variables del programa que las ha invocado.

© 2000-2025, MetaQuotes Ltd.

int _AppliedTo

La variable _AppliedTo permite conocer en el indicador el tipo de datos sobre los que se calcula:

Tipo de datos	Valor	Descripción de los datos usados para el cálculo del indicador
—	0	El indicador usa la segunda forma de llamada OnCalculate(): los datos para el cálculo no se establecen con un solo búfer o matriz de datos
Close	1	Precios Close
Open	2	Precios Open
High	3	Precios High
Low	4	Precios Low
Median Price (HL/2)	5	Precio medio = (High+Low)/2
Typical Price (HLC/3)	6	Precio típico = (High+Low+Close)/3
Weighted Price (HLCC/4)	7	Precio ponderado = (Open+High+Low+Close)/4
Previous Indicator's Data	8	Datos del indicador que ha sido iniciado en el gráfico antes de este indicador
First Indicator's Data	9	Datos del indicador que ha sido iniciado en primer lugar en el gráfico
Indicator handle	10+	Datos del indicador transmitido a la función iCustom() con la ayuda del manejador del indicador. El valor _AppliedTo contiene el manejador del indicador

Ejemplo:

```
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
// obtenemos el tipo de datos sobre el que se calcula el indicador
   Print("_AppliedTo=",_AppliedTo);
   Print(getIndicatorDataDescription(_AppliedTo));
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Descripción de los datos sobre los que se calcula el indicador   |
//+------------------------------------------------------------------+
string getIndicatorDataDescription(int data_id)
  {
   string descr="";
   switch(data_id)
     {
      case(0):descr="It's first type of OnCalculate() - no data buffer";
         break;
      case(1):descr="Indicator calculates on Close price";
         break;
      case(2):descr="Indicator calculates on Open price";
         break;
      case(3):descr="Indicator calculates on High price";
         break;
      case(4):descr="Indicator calculates on Low price";
         break;
      case(5):descr="Indicator calculates on Median Price (HL/2)";
         break;
      case(6):descr="Indicator calculates on Typical Price (HLC/3)";
         break;
      case(7):descr="Indicator calculates on Weighted Price (HLCC/4)";
         break;
      case(8):descr="Indicator calculates Previous Indicator's data";
         break;
      case(9):descr="Indicator calculates on First Indicator's data";
         break;
      default: descr="Indicator calculates on data of indicator with handle="+string(
         break;
     }
//---
   return descr;
  }
```

Vea también

ENUM_APPLIED_PRICE

© 2000-2025, MetaQuotes Ltd.

int _Digits

En la variable _Digits se guarda el número de dígitos después de la coma decimal que determina la precisión de cálculo del precio del símbolo del gráfico corriente.

También se puede usar la función Digits().

© 2000-2025, MetaQuotes Ltd.

double _Point

En la variable _Point se guarda el tamaño del punto de instrumento corriente en divisa de cotización.

También se puede usar la función Point().

© 2000-2025, MetaQuotes Ltd.

int _LastError

En la variable _LastError se guarda el valor del último error ocurrido durante la ejecución del programa mql5. La función ResetLastError() puede reiniciar este valor con el 0.

También se puede usar la función GetLastError() para obtener el código del error.

© 2000-2025, MetaQuotes Ltd.

ENUM_TIMEFRAMES _Period

En la variable _Period se guarda el valor del período de gráfico corriente.

También se puede usar la función Period().

Véase también

PeriodSeconds, Períodos de gráficos, Fecha y hora, Visibilidad de objetos

© 2000-2025, MetaQuotes Ltd.

_RandomSeed

Una variable para guardar el estado actual durante la generación de números pseudoaleatorios enteros. _RandomSeed cambia su valor al llamar a MathRand(). Para establecer el estado inicial necesario, utilice MathSrand().

Un número aleatorio ***x*** que obtiene la función MathRand() se calcula durante cada llamada de la siguiente manera:

```
x=_RandomSeed*214013+2531011;
_RandomSeed=x;
x=(x>>16)&0x7FFF;
```

Véase también

MathRand(), MathSrand(), Tipos enteros

© 2000-2025, MetaQuotes Ltd.

int _StopFlag

La variable _StopFlag almacena la bandera de parada del programa mql5, que será igual a 0 en modo normal de funcionamiento. Cuando el terminal de cliente intente detener el programa, en esta variable se escribirá un valor distinto de cero.

Para comprobar el valor de la bandera _StopFlag también se puede usar la función IsStopped().

© 2000-2025, MetaQuotes Ltd.

string _Symbol

En la variable _Symbol se guarda el nombre del símbolo del gráfico corriente.

También se puede usar la función Symbol().

© 2000-2025, MetaQuotes Ltd.

int _UninitReason

En la variable _UninitReason se guarda el código de la causa de deinicialización del programa.

Suelen obtener el código de la causa de deinicialización usando la función UninitializeReason().

© 2000-2025, MetaQuotes Ltd.

int _IsX64

La variable _IsX64 permite saber en qué terminal se ha iniciado un programa MQL5: para un terminal de 32 bits _IsX64=0 y para un terminal de 64 bits _IsX64!=0.

Asimismo, se puede utilizar la función TerminalInfoInteger(TERMINAL_X64).

Ejemplo:

```
// comprobamos en qué terminal se ha iniciado el programa
   Print("_IsX64=",_IsX64);
   if(_IsX64)
      Print("El programa ",__FILE__," ha sido iniciado en un terminal de 64 bits");
   else
      Print("El programa ",__FILE__," ha sido iniciado en un terminal de 32 bits");
   Print("TerminalInfoInteger(TERMINAL_X64)=",TerminalInfoInteger(TERMINAL_X64));
```

Vea también

MQLInfoInteger, Importación de funciones (#import)

© 2000-2025, MetaQuotes Ltd.

Funciones comunes

Funciones generales que no han entrado en ninguno de los grupos especializados.

Función	Acción
Alert	Muestra el mensaje en una ventana separada
CheckPointer	Devuelve el tipo del puntero a objeto
Comment	Muestra el mensaje en la esquina superior izquierda del gráfico de precios
CryptEncode	Convierte los datos del array fuente en el array receptor según el método especificado
CryptDecode	Conversión inversa de los datos del array
DebugBreak	Punto programado de interrupción en depuración
ExpertRemove	Detiene el trabajo del Asesor Experto y lo descarga del gráfico
GetPointer	Devuelve el puntero a objeto
GetTickCount	Devuelve la cantidad de millisegundos pasados desde el momento del arranque del sistema
GetTickCount64	Devuelve la cantidad de millisegundos pasados desde el momento del arranque del sistema
GetMicrosecondCount	Devuelve la cantidad de milisegundos transcurridos desde el inicio de funcionamiento del programa MQL
MessageBox	Crea y visualiza la ventana de mensajes, además de gestionarlo
PeriodSeconds	Devuelve la cantidad de segundos en el período
PlaySound	Reproduce un archivo de audio
Print	Visualiza un mensaje en el registro
PrintFormat	Formatea e imprime juego de símbolos y valores en un registro histórico de acuerdo con el formato establecido
ResetLastError	Pone el valor de variable predefinida _LastError a cero
ResourceCreate	Esta función crea un recurso de imagen a base de un conjunto de datos
ResourceFree	Elimina el recurso creado dinámicamente (libera la memoria ocupada por el recurso)
ResourceReadImage	Lee los datos del recurso gráfico creado con la función ResourceCreate() o guardado en el archivo EX5 tras la compilación
ResourceSave	Guarda el recurso en el archivo especificado
SetUserError	Pone la variable predefinida _LastError en el valor equivalente a ERR_USER_ERROR_FIRST + user_error

© 2000-2025, MetaQuotes Ltd.

Función	Acción
SetReturnError	Establece el código que retornará el proceso del terminal al completar el trabajo
Sleep	Suspende la ejecución del Asesor Experto en curso o del script por un intervalo determinado
TerminalClose	Envía al terminal el comando de finalizar el trabajo
TesterHideIndicators	Establece el modo de muestra/ocultación de los indicadores que se usan en el experto
TesterStatistics	La función devuelve el valor del indicador estadístico especificado que ha sido calculado a base de los resultados de simulación
TesterStop	Da el comando para finalizar el trabajo del programa durante la simulación
TesterDeposit	Función especial para emular operaciones de depósito de fondos durante la simulación
TesterWithdrawal	Es una función especial para emular las operaciones de retiro de fondos durante el proceso de evaluación
TranslateKey	Retorna un símbolo Unicode según el código virtual de la tecla
ZeroMemory	Reinicializa la variable pasada por referencia. La variable puede ser de cualquier tipo, salvo las clases y estructuras que contienen constructores.

© 2000-2025, MetaQuotes Ltd.

Alert

Visualiza la ventana de diálogo que contiene los datos de usuario.

```
void  Alert(
   argument,      // el primer valor
   ...            // siguientes valores
   );
```

Parámetros

argument

[in] Cualquier valor separado por comas. Para separar la información mostrada en varias líneas se puede usar el símbolo de avance de líneas "\n" o "\r\n". El número de parámetros no puede superar 64.

Valor devuelto

No hay valor devuelto.

Nota

No se puede pasar los arrays a la función Alert(). Los arrays deben visualizarse elemento por elemento. Los datos del tipo double se visualizan con 8 dígitos decimales después del punto, los del tipo float - con 5 dígitos decimales después del punto. Para visualizar los números reales con otra precisión o en formato científico hace falta usar la función DoubleToString().

Los datos del tipo bool se visualizan como cadenas "true" o "false". Las fechas se visualizan como YYYY.MM.DD HH:MI:SS. Para conseguir otro formato de fecha hay que usar la función TimeToString(). Los datos del tipo color se visualizan como cadena R,G,B, o usando el nombre del color si está presente en el juego de colores.

Durante el trabajo en el Probador de Estrategias la función Alert() no se ejecuta.

Ejemplo:

```
//--- enums
enum ENUM_INTERSECT_DIRECTION
  {
   INTERSECT_DIRECTION_NONE= 0,  // no hay cruce
   INTERSECT_DIRECTION_UP  = 1,  // cruce hacia arriba
   INTERSECT_DIRECTION_DOWN=-1,  // cruce hacia abajo
  };

//--- input parameters
input    uint               InpPeriod = 10;            // MA Period
input    int                InpShift  = 0;             // MA Shift
input    ENUM_MA_METHOD     InpMethod = MODE_SMA;      // MA Method
input    ENUM_APPLIED_PRICE InpPrice  = PRICE_CLOSE;   // MA Applied price

//--- global variables
```

© 2000-2025, MetaQuotes Ltd.

```
int      ExtMaHandle;
int      ExtMaPeriod;
double   ExtData[2];
MqlRates ExtRates[2];

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- el periodo de cálculo de la media móvil será igual al valor por defecto (10) si
   ExtMaPeriod=int(InpPeriod<1 ? 10 : InpPeriod);
//--- creamos un manejador del indicador Moving Average con los parámetros especificad
   ExtMaHandle=iMA(Symbol(),PERIOD_CURRENT,ExtMaPeriod,InpShift,InpMethod,InpPrice);
   ResetLastError();
   if(ExtMaHandle==INVALID_HANDLE)
     {
      PrintFormat("Failed to create iMA() handle. Error code: %d",GetLastError());
      return(INIT_FAILED);
     }

//--- obtenemos la hora de la última actualización de precios
   datetime tick_time=TickTime();
//--- obtenemos los datos de la media móvil y los datos de precio de las dos últimas b
   if(GetData(ExtMaHandle,ExtData,ExtRates) && tick_time!=0)
     {
      //--- si el precio está por encima de la media móvil
      if(ExtRates[1].close>ExtData[1])
        {
         //--- creamos un texto de mensaje y mostramos Alert
         string message=StringFormat("Bar time: %s. The price is above the moving aver
         Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SECON
         /*
        Resultado:
         Alert: Bar time: 2024.02.16 18:00. The price is above the moving average at 2
         */
        }
      else
        {
         //--- si el precio está por debajo de la media móvil
         if(ExtRates[1].close<ExtData[1])
           {
            //--- creamos un texto de mensaje y mostramos Alert
            string message=StringFormat("Bar time: %s. The price is below the moving a
            Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SE
            /*
           Resultado:
            Alert: Bar time: 2024.02.16 19:00. The price is below the moving average a
            */
```

© 2000-2025, MetaQuotes Ltd.

```
            }
          else
            {
             //--- creamos un texto de mensaje y mostramos Alert
             string message=StringFormat("Bar time: %s. The price and moving average ar
             Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SE
             /*
            Resultado:
             Alert: Bar time: 2024.02.16 20:00. The price and moving average are equal
             */
            }
        }
     }

//--- con éxito
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   ResetLastError();
//--- obtenemos los datos de la media móvil y los datos de precio de las dos últimas b
   if(!GetData(ExtMaHandle,ExtData,ExtRates))
      return;
//--- obtenemos la dirección en que el precio cruza la media móvil en la barra actual
   ENUM_INTERSECT_DIRECTION intersect=GetIntersectDirection(ExtData,ExtRates);

//--- variable para guardar el mensaje anterior
   static string message_prev="";

//--- si el precio ha cruzado la media móvil en la barra actual de abajo hacia arriba
   if(intersect==INTERSECT_DIRECTION_UP)
     {
      //--- obtenemos la hora del tick en el que se ha producido el cruce
      datetime tick_time=TickTime();
      if(tick_time==0)
         return;
      //--- creamos un texto de mensaje
      string message=StringFormat("Bar time: %s. The price crossed the MA from bottom
      //--- si el mensaje pasado es igual al actual, mostramos Alert con el mensaje y
      if(message!=message_prev)
        {
         Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SECON
         message_prev=message;
         /*
        Resultado:\
         Alert: Bar time: 2024.02.16 09:00. The price crossed the MA from bottom to to
```

© 2000-2025, MetaQuotes Ltd.

```
         */
        }
     }

//--- si el precio ha cruzado la media móvil en la barra actual de arriba hacia abajo
   if(intersect==INTERSECT_DIRECTION_DOWN)
     {
      //--- obtenemos la hora del tick en el que se ha producido el cruce
      datetime tick_time=TickTime();
      if(tick_time==0)
         return;
      //--- creamos un texto de mensaje
      string message=StringFormat("Bar time: %s. The price crossed the MA from top to
      //--- si el mensaje pasado es igual al actual, mostramos Alert con el mensaje y
      if(message!=message_prev)
        {
         Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SECON
         message_prev=message;
         /*
        Resultado:\
         Alert: Bar time: 2024.02.16 10:00. The price crossed the MA from top to botto
         */
        }
     }
  }
//+------------------------------------------------------------------+
//| Obtiene los datos del precio y las medias móviles en arrays      |
//+------------------------------------------------------------------+
bool GetData(int handle,double &ma_data[],MqlRates &price_data[])
  {
   ResetLastError();
//--- obtenemos los datos de la media móvil de las dos últimas barras
   if(CopyBuffer(handle,0,0,2,ma_data)!=2)
     {
      PrintFormat("CopyBuffer() failed. Error code: %d",GetLastError());
      return(false);
     }
//--- obtenemos los datos de precio de las dos últimas barras
   if(CopyRates(Symbol(),PERIOD_CURRENT,0,2,price_data)!=2)
     {
      PrintFormat("CopyRates() failed. Error code: %d",GetLastError());
      return(false);
     }

   return(true);
  }
//+------------------------------------------------------------------+
//| Retorna la dirección en que el precio cruza una media móvil      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
ENUM_INTERSECT_DIRECTION GetIntersectDirection(double &ma_data[],MqlRates &price_data
  {
   double ma0=ma_data[1];
   double ma1=ma_data[0];
   double close0=price_data[1].close;
   double close1=price_data[0].close;

   if(close1<=ma1 && close0>ma0)
      return(INTERSECT_DIRECTION_UP);
   else
     {
      if(close1>=ma1 && close0<ma0)
         return(INTERSECT_DIRECTION_DOWN);
      else
         return(INTERSECT_DIRECTION_NONE);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la hora del tick en segundos                             |
//+------------------------------------------------------------------+
datetime TickTime()
  {
   MqlTick tick={};

   ResetLastError();
   if(!SymbolInfoTick(Symbol(),tick))
     {
      PrintFormat("SymbolInfoTick() failed. Error code: %d",GetLastError());
      return(0);
     }

   return(tick.time);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckPointer

Devuelve el tipo de puntero a objeto.

```
ENUM_POINTER_TYPE  CheckPointer(
   object* anyobject      // puntero a objeto
   );
```

Parámetros

anyobject

[in] Puntero a objeto.

Valor devuelto

Devuelve el valor de enumeración ENUM_POINTER_TYPE.

Nota

El intento de llamar a un puntero incorrecto lleva a la terminación crítica del programa. Por eso es necesario usar la función CheckPointer antes de usar un puntero. Un puntero puede ser incorrecto en las siguientes ocasiones:

- el puntero es igual a NULL;
- si el objeto ha sido eliminado por el operador delete.

Esta función puede ser utilizada para comprobar la validez del puntero. Un valor diferente a cero garantiza que el puntero puede ser utilizado para acceder.

Para comprobar rápidamente el puntero, podemos utilizar también el operador "!"(ejemplo), que comprueba su validez usando una llamada implícita a la función CheckPointer.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Eliminación de la lista mediante eliminación de sus elementos    |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- puntero auxiliar para trabajar en el ciclo
   CItem* item;
//--- pasamos por el ciclo e intentamos eliminar los punteros dinámicos
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Dynamic object ",item.Identifier()," to be deleted");
         delete (item);
        }
      else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
     }
//---
  }
```

Véase también

© 2000-2025, MetaQuotes Ltd.

Punteros a objetos, Comprobación de punteros a objetos, Operador de eliminación de objeto delete

© 2000-2025, MetaQuotes Ltd.

Comment

Visualiza el comentario definido por el usuario en la esquina superior izquierda del gráfico.

```
void  Comment(
   argument,      // el primer valor
   ...            // siguientes valores
   );
```

Parámetros

...

[in] Cualquieres valores separados por comas. Para separar la información mostrada en varias líneas se puede usar el símbolo de avance de líneas "\n" o "\r\n". El número de parámetros no puede superar 64. La longitud total del mensaje a mostrar (inclusive los símbolos auxiliares invisibles) no podrá superar 2045 símbolos (los que sobran van a ser cortados a la hora de ser mostrados).

Valor devuelto

No hay valor devuelto

Nota

No se puede pasar los arrays a la función Comment(). Los arrays tienen que imprimirse elemento por elemento.

Los datos del tipo double se visualizan con la precisión de hasta 16 dígitos decimales después del punto, además, los datos pueden ser visualizados en el formato tradicional o científico, dependiendo de cuál de ellos va a ser más compacto. Los datos del tipo float se visualizan con 5 dígitos decimales después del punto. Para visualizar los números reales con otra precisión o en un formato explícitamente especificado hay que usar la función DoubleToString().

Los datos del tipo bool se visualizan como cadenas "true" o "false". Las fechas se visualizan como YYYY.MM.DD HH:MI:SS. Para conseguir otro formato de fecha hay que usar la función TimeToString(). Los datos del tipo color se visualizan como cadena R,G,B, o usando el nombre del color si está presente en el juego de colores.

Durante el trabajo en el Probador de Estrategias en el modo de optimización, la función Comment() no se ejecuta.

Ejemplo:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
//--- Mostremos los valores en tres líneas
   Comment(StringFormat("Mostramos precios\nAsk = %G\nBid = %G\nSpread = %d",Ask,Bid,S
  }
```

© 2000-2025, MetaQuotes Ltd.

Véase también

ChartSetString, ChartGetString

© 2000-2025, MetaQuotes Ltd.

CryptEncode

Convierte los datos del array fuente en el array receptor según el método especificado.

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,        // método de conversión
   const uchar&        data[],        // array fuente
   const uchar&        key[],         // clave de cifrado
   uchar&              result[]       // array receptor
   );
```

Parámetros

method

[in] Método de conversión. Puede ser uno de los valores de la enumeración ENUM_CRYPT_METHOD.

data[]

[in] Array fuente.

key[]

[in] Clave de cifrado.

result[]

[out] Array receptor.

Valor devuelto

Número de bytes en el array receptor, o 0 en caso del error. Para obtener la información adicional sobre el error, hay que llamar a la función GetLastError().

Ejemplo:

```
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";
//--- comprobando el tamaño
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);
//--- convirtiendo en una cadena hexadecimal
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   string text="The quick brown fox jumps over the lazy dog";
   string keystr="ABCDEFG";
   uchar src[],dst[],key[];
//--- preparando la clave de cifrado
   StringToCharArray(keystr,key);
//--- preparando el array de origen src[]
   StringToCharArray(text,src);
//--- mostrando los datos de origen
   PrintFormat("Initial data: size=%d, string='%s'",ArraySize(src),CharArrayToString(s
//--- cifrando el array src[] con el método DES con una clave key[] de 56 bits
   int res=CryptEncode(CRYPT_DES,src,key,dst);
//--- comprobando los resultados del cifrado
   if(res>0)
     {
      //--- mostrando los datos cifrados
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
      //--- descifrando el array dst[] con el método DES con una clave key[] de 56 bit
      res=CryptDecode(CRYPT_DES,dst,key,src);
     //--- comprobando el resultado
      if(res>0)
        {
         //--- mostrando los datos descifrados
         PrintFormat("Decoded data: size=%d, string='%s'",ArraySize(src),CharArrayToSt
        }
      else
         Print("Error en CryptDecode. Código de error=",GetLastError());
     }
   else
      Print("Error en CryptEncode. Código de error=",GetLastError());
  }
```

Véase también

Operaciones con arrays, CryptDecode()

© 2000-2025, MetaQuotes Ltd.

CryptDecode

Conversión inversa de los datos del array obtenido a través de la función CryptEncode().

```
int  CryptDecode(
   ENUM_CRYPT_METHOD   method,        // método de conversión
   const uchar&        data[],        // array fuente
   const uchar&        key[],         // clave de cifrado
   uchar&              result[]       // array receptor
   );
```

Parámetros

method

[in] Método de conversión. Puede ser uno de los valores de la enumeración ENUM_CRYPT_METHOD.

data[]

[in] Array fuente.

key[]

[in] Clave de cifrado.

result[]

[out] Array receptor.

Valor devuelto

Número de bytes en el array receptor, o 0 en caso del error. Para obtener la información adicional sobre el error, hay que llamar a la función GetLastError().

Ejemplo:

```
input string InpKey = "ABCDEFG";  // Encryption key

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string text="The quick brown fox jumps over the lazy dog";
   uchar src[],dst[],key[];

//--- preparando la clave de cifrado
   StringToCharArray(InpKey,key);
//--- preparando el array de origen src[]
   StringToCharArray(text,src);
//--- mostrando los datos de origen
   PrintFormat("Initial data: size=%d, string='%s'",ArraySize(src),CharArrayToString(s
```

© 2000-2025, MetaQuotes Ltd.

```
//--- cifrando el array src[] con el método DES con una clave key[] de 56 bits
   int res=CryptEncode(CRYPT_DES,src,key,dst);

//--- comprobando los resultados del cifrado
   if(res>0)
     {
      //--- mostrando los datos cifrados
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
      //--- descifrando el array dst[] con el método DES con una clave key[] de 56 bit
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- comprobamos el resultado
      if(res>0)
        {
         //--- mostrando los datos descifrados
         PrintFormat("Decoded data: size=%d, string='%s'",ArraySize(src),CharArrayToSt
        }
      else
         Print("CryptDecode failed. Error: ",GetLastError());
     }
   else
      Print("CryptEncode failed. Error: ",GetLastError());
  }
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";

//--- comprobando el tamaño
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);

//--- convirtiendo en una cadena hexadecimal
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);

   return(res);
  }
```

Véase también

Operaciones con arrays, CryptEncode()

© 2000-2025, MetaQuotes Ltd.

DebugBreak

Es el punto programado de interrupción en depuración.

```
void  DebugBreak();
```

Valor devuelto

No hay valor devuelto.

Nota

La ejecución de un programa mql5 se interrumpe sólo si el programa está inicializado en modo de depuración. Se puede utilizar esta función para ver los valores de las variables y/o la siguiente ejecución paso a paso.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- compilar el archivo con F5
//--- en el modo de depuración, con i == j nos detendremos en la línea DebugBreak()
   for(int i=0,j=20; i<20; i++,j--)
     {
      if(i==j)
         DebugBreak();
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

ExpertRemove

Detiene el trabajo del Asesor Experto y lo descarga del gráfico.

```
void  ExpertRemove();
```

Valor devuelto

No hay valor devuelto.

Nota

El Asesor Experto no se detiene inmediatamente si se llama a la función ExpertRemove(), únicamente se activa la bandera para detener la ejecución del Asesor Experto. Es decir, el Asesor Experto no va a procesar ninguno de los siguientes eventos, se invocará la función OnDeinit() y El Asesor Experto será descargado y borrado del gráfico.

La llamada de ExpertRemove() en el simulador de estrategias dentro del manejador OnInit() conllevará la cancelación de la simulación en el actual conjunto de parámetros. Esta finalización se considera un error de inicialización.

Al llamar ExpertRemove() en el simulador de estrategias después de inicializar el asesor con éxito, la simulación finalizará de la forma normal, con la llamada de OnDeinit() y OnTester(). En este caso, se obtendrán todas las estadísticas comerciales y el valor del criterio de optimización.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              Test_ExpertRemove.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
input int ticks_to_close=20;// número de ticks antes de descargar el Asesor Experto
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(TimeCurrent(),": " ,__FUNCTION__,"reason code = ",reason);
//--- "clear" comment
   Comment("");
//---
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   static int tick_counter=0;
//---
   tick_counter++;
   Comment("\nHasta la descarga del Asesor Experto ",__FILE__," quedan",
           (ticks_to_close-tick_counter)," ticks");
//--- hasta
   if(tick_counter>=ticks_to_close)
     {
      ExpertRemove();
      Print(TimeCurrent(),": ",__FUNCTION__," Asesor Experto será descargado");
     }
   Print("tick_counter = ",tick_counter);
//---
  }
//+------------------------------------------------------------------+
```

Véase también

Funcionamiento de programas, Eventos del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

GetPointer

Devuelve el puntero a objeto.

```
void*  GetPointer(
   any_class anyobject      // objeto de cualquier clase
   );
```

Parámetros

anyobject

[in] Objeto de cualquier clase.

Valor devuelto

La función devuelve el puntero a objeto.

Nota

Sólo los objetos de clases tienen punteros. Las instancias de estructuras y las variables de tipos simples no tienen punteros. Un objeto de clase que no ha sido creado mediante el operador new(), sino, por ejemplo, ha sido creado automáticamente en el array de objetos, igualmente tiene un puntero. Pero este puntero va a ser del tipo automático POINTER_AUTOMATIC, y no se le puede aplicar el operador delete(). Aparte de eso, este puntero del tipo no tendrá ninguna diferencia de los punteros dinámicos del tipo POINTER_DYNAMIC.

Debido a que las variables del tipo de estructuras y de tipos simples no tienen punteros, está prohibido aplicarles la función GetPointer(). También está prohibido pasar el puntero como argumento de la función. En todos los casos mencionados el compilador avisará sobre un error.

El intento de llamar a un puntero incorrecto lleva a la terminación crítica del programa, con lo cual es necesario usar la función CheckPointer() antes de utilizar un puntero. Un puntero puede ser incorrecto en las siguientes ocasiones:

- el puntero es igual a NULL;
- si el objeto ha sido eliminado por el operador delete.

Esta función puede ser utilizada para comprobar la validez del puntero. Un valor diferente a cero garantiza que el puntero puede ser utilizado para acceder.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                Check_GetPointer.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

//+------------------------------------------------------------------+
//|  clase que implementa el elemento de la lista                    |
//+------------------------------------------------------------------+
class CItem
  {
   int               m_id;
   string            m_comment;
   CItem*            m_next;
public:
                     CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
                    ~CItem() { Print("Destructor of ",m_id,
                                     (CheckPointer(GetPointer(this))==POINTER_DYNAMIC)
                                     "dynamic":"non-dynamic"); }
   void              Initialize(int id,string comm) { m_id=id; m_comment=comm; }
   void              PrintMe() { Print(__FUNCTION__,":",m_id,m_comment); }
   int               Identifier() { return(m_id); }
   CItem*            Next() {return(m_next); }
   void              Next(CItem *item) { m_next=item; }
  };
//+------------------------------------------------------------------+
//|  la clase más simple de la lista                                 |
//+------------------------------------------------------------------+
class CMyList
  {
   CItem*            m_items;
public:
                     CMyList() { m_items=NULL; }
                    ~CMyList() { Destroy(); }
    bool             InsertToBegin(CItem* item);
    void             Destroy();
  };
//+------------------------------------------------------------------+
//|  inserción del elemento de la lista al principio de todo         |
//+------------------------------------------------------------------+
bool CMyList::InsertToBegin(CItem* item)
  {
   if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
   item.Next(m_items);
   m_items=item;
//---
   return(true);
  }
//+------------------------------------------------------------------+
//|  eliminación de la lista mediante eliminación de sus elementos   |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- puntero auxiliar para trabajar en el ciclo
   CItem* item;
//--- pasamos por el ciclo e intentamos eliminar los punteros dinámicos
   while(CheckPointer(m_items)!=POINTER_INVALID)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       item=m_items;
       m_items=m_items.Next();
       if(CheckPointer(item)==POINTER_DYNAMIC)
         {
          Print("Dynamyc object ",item.Identifier()," to be deleted");
          delete (item);
         }
       else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
      }
//---
  }
//+------------------------------------------------------------------+
//|  Script program start function                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
   CMyList list;
   CItem   items[10];
   CItem*  item;
//--- creamos y añadimos a la lista un puntero dinámico a objeto
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(100,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- añadimos los punteros automáticos a la lista
   for(int i=0; i<10; i++)
     {
      items[i].Initialize(i,"automatic");
      items[i].PrintMe();
      item=GetPointer(items[i]);
      if(CheckPointer(item)!=POINTER_INVALID)
         list.InsertToBegin(item);
     }
//--- añadimos otro puntero dinámico a objeto al principio de la lista
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(200,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- eliminamos los elementos de la lista
   list.Destroy();
//--- todos los elementos de la lista van a ser borrados,
//--- véase la pestaña Experts en el terminal
  }
```

Véase también

Punteros a objetos, Comprobación de puntero a objeto, Operador de eliminación de objeto delete

© 2000-2025, MetaQuotes Ltd.

GetTickCount

La función GetTickCount() devuelve la cantidad de milisegundos pasados desde el momento de arranque del sistema.

```
uint  GetTickCount();
```

Valor devuelto

Valor del tipo uint.

Nota

El contador está limitado por el poder resolutivo del temporizador de sistema. El tiempo se almacena como un entero sin signo, por eso se sobrellena cada 49.7 días con el funcionamiento interrumpido del ordenador.

Ejemplo:

```
#define MAX_SIZE 40
//+------------------------------------------------------------------+
//| un script para medir el tiempo de calculaciones de 40 números de Fibonacci
//+------------------------------------------------------------------+
void OnStart()
  {
//--- recordemos el valor inicial
   uint start=GetTickCount();
//--- variable para obtener el siguiente número de la serie de Fibonacci
   long fib=0;
//--- ciclo en el que se calcula la cantidad especificada de los números de la serie
   for(int i=0;i<MAX_SIZE;i++) fib=TestFibo(i);
//--- obtenemos el tiempo tardado en millisegundos
   uint time=GetTickCount()-start;
//--- mostramos el mensaje en el diario "Asesores Expertos"
   PrintFormat("El cálculo de %d primeros números de Fibonacci ha requerido %d ms",MAX
//--- el trabajo del script está finalizado
   return;
  }
//+------------------------------------------------------------------+
//| La función para obtener el número de Fibonacci según su número ordinal
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- el primer miembro de la serie de Fibonacci
   if(n<2) return(1);
//--- todos los demás miembros se calculan según la fórmula que viene a continuación
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

Véase también

Fecha y hora, EventSetMillisecondTimer, GetTickCount64, GetMicrosecondCount

© 2000-2025, MetaQuotes Ltd.

GetTickCount64

La función GetTickCount()64 retorna el número de milisegundos transcurridos desde el inicio del sistema.

```
ulong  GetTickCount64();
```

Valor retornado

Valor del tipo ulong.

Observación

El contador está limitado por la resolución del temporizador del sistema, que normalmente retorna el resultado con una precisión de 10-16 milisegundos. A diferencia de la función GetTickCount, que tiene el tipo uint y por eso se satura cada 49.7 días si la computadora funciona constantemente, GetTickCount64() puede usarse indefinidamente en lo que respecta al funcionamiento de la computadora, y sin temer además a la saturación.

Ejemplo:

```
#define MAX_SIZE 40

//+------------------------------------------------------------------+
//| Script que mide el tiempo de cálculo de 40 números de Fibonacci  |
//+------------------------------------------------------------------+
void OnStart()
  {
   long fib_array[MAX_SIZE];

//--- recordando el valor inicial
   ulong start=GetTickCount64();
//--- ciclo en el que calculamos la cantidad especificada de números de la serie de Fi
   for(int i=0;i<MAX_SIZE;i++)
      fib_array[i]=TestFibo(i);
//--- obtenemos el tiempo empleado en milisegundos
   ulong time=GetTickCount64()-start;

//--- mostramos en el diario "Expertos" el mensaje
   ArrayPrint(fib_array);
   PrintFormat("Calculating the first %d Fibonacci numbers took %I64u ms",MAX_SIZE,tir
  }
//+------------------------------------------------------------------+
//| Función para obtener un número de Fibonacci según su ordinal     |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- primer miembro de la serie de Fibonacci
   if(n<2)
      return(1);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- todos los términos subsiguientes se calculan mediante esta fórmula
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

Ver también

Fecha y hora, EventSetMillisecondTimer, GetTickCount, GetMicrosecondCount

© 2000-2025, MetaQuotes Ltd.

GetMicrosecondCount

La función GetMicrosecondCount() devuelve la cantidad de microsegundos transcurridos desde el inicio de funcionamiento del programa MQL5.

```
ulong  GetMicrosecondCount();
```

Valor devuelto

Valor del tipo ulong.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Código a probar                                                  |
//+------------------------------------------------------------------+
void Test()
  {
   int    res_int=0;
   double res_double=0;
//---
   for(int i=0;i<10000;i++)
     {
      res_int+=i*i;
      res_int++;
      res_double+=i*i;
      res_double++;
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   uint   ui=0,ui_max=0,ui_min=INT_MAX;
   ulong  ul=0,ul_max=0,ul_min=INT_MAX;
//--- número de pruebas
   for(int count=0;count<1000;count++)
     {
      uint  ui_res=0;
      ulong ul_res=0;
      //---
      for(int n=0;n<2;n++)
        {
         //--- seleccionamos modo de medición
         if(n==0) ui=GetTickCount();
         else     ul=GetMicrosecondCount();
         //--- código a probar
         Test();
         //--- acumulamos el resultado de medición (dependiendo del modo)
```

© 2000-2025, MetaQuotes Ltd.

```
         if(n==0) ui_res+=GetTickCount()-ui;
         else     ul_res+=GetMicrosecondCount()-ul;
        }
      //--- apuntamos el tiempo mínimo y máximo de ejecución del código de ambas dimen
      if(ui_min>ui_res) ui_min=ui_res;
      if(ui_max<ui_res) ui_max=ui_res;
      if(ul_min>ul_res) ul_min=ul_res;
      if(ul_max<ul_res) ul_max=ul_res;
     }
//---
   Print("GetTickCount error(msec): ",ui_max-ui_min);
   Print("GetMicrosecondCount error(msec): ",DoubleToString((ul_max-ul_min)/1000.0,2))
  }
```

Véase también

Fecha y hora, GetTickCount, GetTickCount64

© 2000-2025, MetaQuotes Ltd.

MessageBox

Crea y visualiza la ventana de mensajes, también lo gestiona. La ventana de mensajes contiene un mensaje y encabezamiento, cualquier combinación de signos predefinidos y botones de dirección.

```
int  MessageBox(
   string  text,              // texto del mensaje
   string  caption=NULL,      // encabezamiento de la ventana
   int     flags=0            // define la combinación de botones en la ventana
   );
```

Parámetros

text

[in] Texto que contiene mensaje para visualizar.

caption=NULL

[in] Texto opcional para visualizar en el encabezamiento de la ventana del mensaje. Si este parámetro está vacío, en el encabezamiento de la ventana se mostrará el nombre del Asesor Experto.

flags=0

[in] Banderas opcionales que determinan la apariencia y comportamiento de la ventana de diálogo. Las banderas pueden ser una combinación de un grupo especial de banderas.

Valor devuelto

Si la función se ejecuta con éxito, el valor devuelto es uno de los valores del código de devolución MessageBox().

Nota

La función está prohibida en los indicadores de usuario, pues la llamada de MessageBox() interrumpe el funcionamiento del flujo de ejecución mientras se espera la respuesta del usuario. Y puesto que todos los indicadores de cada símbolo se ejecutan en un único flujo, semejante interrupción hará imposible el funcionamiento de todos los gráficos en todos los marcos temporales de este símbolo.

Durante el trabajo en el Probador de Estrategias la función MessageBox() no se ejecuta.

Ejemplo:

```
//+------------------------------------------------------------------+
//| El asesor muestra MessageBox solicitando el trabajo subsiguiente |
//| al alcanzar el número indicado  de transacciones con pérdidas    |
//| Esperamos el número de barras indicado y mostramos MessageBox    |
//| solicitando el trabajo subsiguiente.                             |
//| Para comprobar, basta abrir y cerrar manualmente con pérdidas    |
//| varias posiciones, ya que para simplificar, el asesor no controla|
//| "sus" posiciones según el número mágico.                         |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//--- input parameters
input uint InpMaxLossDeals      =  3;    // Max Loss deals
input uint InpInactivityNumBars =  5;    // Number of bars of advisor inactivity

//--- global variables
bool     ExtFirstStart=true;             // Bandera de primer inicio
bool     ExtFlag=true;                   // Bandera de permiso de funcionamiento del a
uint     ExtNumLoss;                     // Número de transacciones perdedoras consecu
datetime ExtTimeLastLoss;                // Hora de la última operación de cierre de u

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- obtenemos el número de transacciones perdedoras consecutivas y la hora de la últ
   ExtNumLoss=GetNumLosingTradesInRow(ExtTimeLastLoss);

   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   Comment("");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- determinamos cuántas barras han pasado desde la última posición perdedora cerrad
   int bars_remaining=iBarShift(Symbol(),PERIOD_CURRENT,ExtTimeLastLoss);

//--- si es el primer inicio
   if(ExtFirstStart)
     {
      //--- si el número especificado de barras ya ha pasado después de una serie de p
      if(bars_remaining>(int)InpInactivityNumBars)
         ExtFlag=true;
      ExtFirstStart=false;
     }

//--- si la bandera de funcionamiento del asesor se ha desactivado
   if(!ExtFlag)
     {
      Comment(StringFormat("The advisor is stopped for %d bars. Num Loss positions: %u
```

© 2000-2025, MetaQuotes Ltd.

```
                              (InpInactivityNumBars-bars_remaining),ExtNumLoss,TimeToStrin
      //--- si ha transcurrido el número de barras especificado tras una serie de posi
      if(bars_remaining>(int)InpInactivityNumBars)
        {
         //--- mostramos la ventana MessageBox con el texto y el encabezado de ventana
         //--- la ventana de solicitud tiene dos botones Yes/No y un icono con un sign
         //--- el botón Yes está seleccionado por defecto.
         string mb_text="The specified number of bars of EA inactivity have passed.\n
         string mb_caption="Please note";
         int    mb_id=MessageBox(mb_text,mb_caption,MB_YESNO|MB_ICONQUESTION|MB_DEFBUT
         //--- si el código de retorno de MessageBox es un botón Yes pulsado, establec
         if(mb_id==IDYES)
           {
            ExtFlag=true;
            return;
           }
        }
      //--- la bandera del asesor está desactivada, salimos de OnTick()
      return;
     }

//---  la bandera de funcionamiento del asesor está activada, el asesor funciona como
   Comment(StringFormat("The advisor is working. Num Loss positions: %u, Time last los
           ExtNumLoss,TimeToString(ExtTimeLastLoss,TIME_DATE|TIME_MINUTES|TIME_SECONDS
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction& trans,
                        const MqlTradeRequest& request,
                        const MqlTradeResult& result)
  {
//--- si el tipo de transacción es la adición de una operación a la historia
   if(trans.type==TRADE_TRANSACTION_DEAL_ADD)
     {
      //--- obtenemos el ticket de la transacción y seleccionamos una transacción de l
      ulong deal_ticket=trans.deal;
      if(HistoryDealSelect(deal_ticket))
        {
         //--- si se trata de una transacción de salida del mercado, obtenemos el núme
         ENUM_DEAL_ENTRY entry=(ENUM_DEAL_ENTRY)HistoryDealGetInteger(deal_ticket,DEAL
         if(entry==DEAL_ENTRY_OUT || entry==DEAL_ENTRY_INOUT || entry==DEAL_ENTRY_OUT_
            ExtNumLoss=GetNumLosingTradesInRow(ExtTimeLastLoss);
        }
     }

//--- si el número de transacciones perdedoras consecutivas es superior al valor espec
   if(ExtNumLoss>=InpMaxLossDeals && ExtFlag)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- mostramos la ventana MessageBox con el texto y el encabezado de ventana ir
      //--- la ventana de solicitud tiene dos botones Yes/No y un icono con un signo d
      //--- el botón No está seleccionado por defecto.
      string mb_text="The number of losing trades has reached the specified maximum. T
      string mb_caption="Attention!";
      int    mb_id=MessageBox(mb_text,mb_caption,MB_YESNO|MB_ICONQUESTION|MB_DEFBUTTON
      //--- si el código de retorno de MessageBox es un botón No pulsado, quitamos la
      if(mb_id==IDNO)
         ExtFlag=false;
     }
  }
//+------------------------------------------------------------------+
//| Retorna el número de transacciones perdedoras consecutivas       |
//| y la hora de la última operación de cierre                       |
//| de una posición perdedora                                        |
//+------------------------------------------------------------------+
uint GetNumLosingTradesInRow(datetime &time_last_deal)
  {
//--- Seleccionamos la historia completa
   if(!HistorySelect(0,TimeCurrent()))
      return(0);

//--- en un ciclo a través de la lista de transacciones históricas obtenemos el ticket
   uint res=0;
   uint total=HistoryDealsTotal();
   for(int i=(int)total-1; i>=0; i--)
     {
      ulong deal_ticket=HistoryDealGetTicket(i);
      if(deal_ticket>0)
        {
         //--- si la transacción no es de salida de la posición, pasamos a la siguient
         ENUM_DEAL_ENTRY entry=(ENUM_DEAL_ENTRY)HistoryDealGetInteger(deal_ticket,DEAL
         if(entry!=DEAL_ENTRY_OUT && entry!=DEAL_ENTRY_OUT_BY && entry!=DEAL_ENTRY_INO
            continue;
         //--- si el resultado del cierre de posición tiene un beneficio positivo, int
         if(!IsClosePositionWithLoss(deal_ticket))
            break;
         //--- aumentamos el contador de transacciones consecutivas con beneficio nega
         res++;
         //--- escribimos en la variable la hora máxima de la transacción (buscando la
         datetime deal_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
         if(deal_time>time_last_deal)
            time_last_deal=deal_time;
        }
     }

//--- retorna el número de pérdidas consecutivas
   return(res);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Retorna la bandera de cierre de una posición con pérdidas        |
//+------------------------------------------------------------------+
bool IsClosePositionWithLoss(const ulong deal_ticket)
  {
//--- obtenemos de la transacción los valores de las propiedades que afectan a los ber
   double profit=HistoryDealGetDouble(deal_ticket,DEAL_PROFIT);
   double comission=HistoryDealGetDouble(deal_ticket,DEAL_COMMISSION);
   double swap=HistoryDealGetDouble(deal_ticket,DEAL_SWAP);
   double fee=HistoryDealGetDouble(deal_ticket,DEAL_FEE);

//--- retorna la bandera que indica que el valor total de las propiedades obtenidas es
   return(profit+comission+swap+fee<0);
  }
```

© 2000-2025, MetaQuotes Ltd.

PeriodSeconds

Devuelve la cantidad de segundos en el período.

```
int  PeriodSeconds(
   ENUM_TIMEFRAMES  period=PERIOD_CURRENT      // período del gráfico
   );
```

Parámetros

period=PERIOD_CURRENT

[in] Valor del período de gráfico de la enumeración ENUM_TIMEFRAMES. Si el parámetro no está especificado, se devuelve el número de segundos del período actual del gráfico en el cual el programa está inicializado.

Valor devuelto

Número de segundos en el período especificado.

Ejemplo:

```
#include <Trade\Trade.mqh>
#define MAGIC (123)

//--- input parameters
input string          InpFileNameOK  = "ok.wav";               // archivo con el sonic
input string          InpFileNameErr = "timeout.wav";          // archivo con el sonic
input ENUM_ORDER_TYPE InpOrderType   = ORDER_TYPE_BUY_LIMIT;   // tipo de orden
input double          InpLots        = 0.1;                    // lotes

//--- global variables
CTrade ExtTrade;

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- establecemos el número mágico y el tipo de orden por ejecución según la configu
   ExtTrade.SetExpertMagicNumber(MAGIC);
   ExtTrade.SetTypeFillingBySymbol(Symbol());
//--- llamamos a la función para colocar una orden o abrir una posición con reproducci
   OrderSendWithAudio();
  }
//+------------------------------------------------------------------+
//| La función coloca una orden o bien abre una posición             |
//| y reproduce un sonido de éxito o error                           |
//+------------------------------------------------------------------+
void OrderSendWithAudio(void)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   bool     res=true;
   MqlTick tick= {};

   ResetLastError();
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      PlaySound(InpFileNameErr);
      return;
     }
//--- enviamos una solicitud al servidor
   switch(InpOrderType)
     {
      case ORDER_TYPE_BUY :
         res=ExtTrade.Buy(InpLots);
         break;
      case ORDER_TYPE_BUY_LIMIT :
         res=ExtTrade.BuyLimit(InpLots,NormalizeDouble(tick.ask-100*Point(),Digits()))
         break;
      case ORDER_TYPE_BUY_STOP :
         res=ExtTrade.BuyStop(InpLots,NormalizeDouble(tick.ask+100*Point(),Digits()));
         break;
      case ORDER_TYPE_SELL :
         res=ExtTrade.Sell(InpLots);
         break;
      case ORDER_TYPE_SELL_LIMIT :
         res=ExtTrade.SellLimit(InpLots,NormalizeDouble(tick.bid+100*Point(),Digits())
         break;
      case ORDER_TYPE_SELL_STOP :
         res=ExtTrade.SellStop(InpLots,NormalizeDouble(tick.bid-100*Point(),Digits()))
         break;
      default :
         res=false;
     }
   if(!res)
      Print("Error ",GetLastError());
   Print(ExtTrade.ResultRetcodeDescription());

//--- si la solicitud ha sido aceptada, reproducimos el sonido ok.wav
   if(ExtTrade.ResultRetcode()==TRADE_RETCODE_DONE)
      PlaySound(InpFileNameOK);
   else
      PlaySound(InpFileNameErr);
  }
```

Véase también

_Period, Períodos de gráficos, Fecha y hora, Visibilidad de objetos

© 2000-2025, MetaQuotes Ltd.

PlaySound

Reproduce archivo de audio.

```
bool  PlaySound(
   string  filename      // nombre del archivo
   );
```

Parámetros

filename

[in] La ruta del archivo de audio. Si filename=NULL, la reproducción del sonido se para.

Valor devuelto

true - si el archivo de audio ha sido encontrado, en caso contrario devuelve false.

Nota

El archivo tiene que estar ubicado en la carpeta directorio_de_terminal\Sounds o en su subcarpeta. Se reproducen sólo los archivos de audio en el formato WAV.

La llamada a PlaySound() con el parámetro NULL para la reproducción del sonido.

Durante el trabajo en el Probador de Estrategias la función PlaySound() no se ejecuta.

Ejemplo:

```
#include <Trade\Trade.mqh>
#define MAGIC (123)

//--- input parameters
input string          InpFileNameOK  = "ok.wav";               // Success sound Filena
input string          InpFileNameErr = "timeout.wav";          // Error sound Filename
input ENUM_ORDER_TYPE InpOrderType   = ORDER_TYPE_BUY_LIMIT;   // Order type
input double          InpLots        = 0.1;                    // Lots

//--- global variables
CTrade ExtTrade;

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- establecemos el número mágico y el tipo de orden por ejecución según la configu
   ExtTrade.SetExpertMagicNumber(MAGIC);
   ExtTrade.SetTypeFillingBySymbol(Symbol());
//--- llamamos a la función para colocar una orden o abrir una posición con reproducci
   OrderSendWithAudio();
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| La función coloca una orden o abre una posición                  |
//| y reproduce un sonido de éxito o error                           |
//+------------------------------------------------------------------+
void OrderSendWithAudio(void)
  {
   bool    res=true;
   MqlTick tick= {};

   ResetLastError();
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      PlaySound(InpFileNameErr);
      return;
     }
//--- enviamos una solicitud al servidor
   switch(InpOrderType)
     {
      case ORDER_TYPE_BUY :
         res=ExtTrade.Buy(InpLots);
         break;
      case ORDER_TYPE_BUY_LIMIT :
         res=ExtTrade.BuyLimit(InpLots,NormalizeDouble(tick.ask-100*Point(),Digits()))
         break;
      case ORDER_TYPE_BUY_STOP :
         res=ExtTrade.BuyStop(InpLots,NormalizeDouble(tick.ask+100*Point(),Digits()));
         break;
      case ORDER_TYPE_SELL :
         res=ExtTrade.Sell(InpLots);
         break;
      case ORDER_TYPE_SELL_LIMIT :
         res=ExtTrade.SellLimit(InpLots,NormalizeDouble(tick.bid+100*Point(),Digits())
         break;
      case ORDER_TYPE_SELL_STOP :
         res=ExtTrade.SellStop(InpLots,NormalizeDouble(tick.bid-100*Point(),Digits()))
         break;
      default :
         res=false;
     }
   if(!res)
      Print("Error ",GetLastError());
   Print(ExtTrade.ResultRetcodeDescription());

//--- si la solicitud ha sido aceptada, reproducimos el sonido ok.wav
   if(ExtTrade.ResultRetcode()==TRADE_RETCODE_DONE)
      PlaySound(InpFileNameOK);
   else
      PlaySound(InpFileNameErr);
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

Véase también

Recursos

© 2000-2025, MetaQuotes Ltd.

Print

Imprime un mensaje en el registro del Asesor Experto. Los parámetros pueden tener cualquier tipo.

```
void  Print(
   argument,      // el primer valor
   ...            // siguientes valores
   );
```

Parámetros

...

[in] Cualquieres valores separados por comas. El número de parámetros no puede superar 64.

Nota

No se puede pasar los arrays a la función Print(). Los arrays deben imprimirse elemento por elemento.

Los datos del tipo double se visualizan con la precisión de hasta 16 dígitos decimales después del punto, además, los datos pueden ser visualizados en el formato tradicional o científico, dependiendo de cuál de ellos va a ser más compacto. Los datos del tipo float se visualizan con 5 dígitos decimales después del punto. Para visualizar los números reales con otra precisión o en un formato explícitamente especificado hay que usar la función PrintFormat().

Los datos del tipo bool se visualizan como cadenas "true" o "false". Las fechas se visualizan como YYYY.MM.DD HH:MI:SS. Para conseguir otro formato de fecha hay que usar la función TimeToString(). Los datos del tipo color se visualizan como cadena R,G,B, o usando el nombre del color si está presente en el juego de colores.

Durante el trabajo en el Probador de Estrategias en el modo de optimización, la función Print() no se ejecuta.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- imprimimos DBL_MAX utilizando Print(), esto equivale a PrintFormat(%%.16G,DBL_MA
   Print("---- como se ve DBL_MAX -----");
   Print("Print(DBL_MAX)=",DBL_MAX);
//--- ahora imprimimos el número DBL_MAX utilizando PrintFormat()
   PrintFormat("PrintFormat(%%.16G,DBL_MAX)=%.16G",DBL_MAX);
//--- Impresión en el diario "Asesores Expertos"
// Print(DBL_MAX)=1.797693134862316e+308
// PrintFormat(%.16G,DBL_MAX)=1.797693134862316E+308

//--- vamos a ver cómo se imprime el tipo float
   float c=(float)M_PI; // hay que convertir explícitamente al tipo de meta
   Print("c=",c, "    Pi=",M_PI, "    (float)M_PI=",(float)M_PI);
// c=3.14159    Pi=3.141592653589793    (float)M_PI=3.14159

//--- mostraremos lo que puede pasar durante las operaciones aritméticas con los tipos
   double a=7,b=200;
   Print("---- antes de las operaciones aritméticas");
   Print("a=",a,"   b=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- vamos a dividir a por b (7/200)
   a=a/b;
//--- ahora como si hubieramos restaurado el valor en la variable b
   b=7.0/a; // se espera que b=7.0/(7.0/200.0)=>7.0/7.0*200.0=200 - pero eso no es así
//--- vamos a imprimir otra vez el valor calculado b
   Print("----- después de las operaciones aritméticas");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- impresión en el diario "Asesores Expertos"
// Print(b)=200.0
// Print(DoubleToString(b,16))=199.9999999999999716 (vemos que en realidad b ya no es

//--- vamos a crear un valor muy pequeño epsilon=1E-013
   double epsilon=1e-13;
   Print("---- vamos a crear un número muy pequeño");
   Print("epsilon=",epsilon); // obtenemos   epsilon=1E-013
//--- ahora restamos el épsilon del número b e imprimimos otra vez el valor en el diar
   b=b-epsilon;
//--- imprimimos de dos maneras
   Print("---- después de la sustracción de epsilon de la variable b");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- impresión en el diario "Expertos"
// Print(b)=199.9999999999999  (ahora el valor b después de la sustracción del épsilon
// Print(DoubleToString(b,16))=199.9999999999998578
//    (ahora el valor b después de la sustracción del épsilon no puede ser redondeado
  }
```

Véase también

DoubleToString, StringFormat

© 2000-2025, MetaQuotes Ltd.

PrintFormat

Formatea e imprime juegos de símbolos y valores en el registro histórico del Asesor Experto conforme al formato predefinido.

```
void  PrintFormat(
   string format_string,    // línea de formato
   ...                      // valor de tipos simples
   );
```

Parámetros

format_string

[in] Una línea de formato se compone de símbolos simples y si los argumentos siguen la línea de formato, también contiene la especificación de formato.

...

[in] Cualquieres valores de tipos simples separados por comas. El número total de parámetros no puede superar 64, inclusive la línea de formato.

Valor devuelto

Una cadena.

Nota

Durante el trabajo en el Probador de Estrategias en el modo de optimización, la función PrintFormat() no se ejecuta.

La cantidad, orden y tipo de parámetros tiene que corresponder exactamente a la composición de los especificadores, en caso contrario el resultado de la impresión estará indefinida. En vez de la función PrintFormat() se puede usar la función printf().

Si después de la línea de formato siguen los parámetros, esta línea debe contener las especificaciones del formato que denotan el formato de salida de estos parámetros. La especificación del formato siempre se empieza con el signo del porcentaje (%).

Una línea de formato se lee de izquierda a derecha. Al encontrar la primera especificación del formato (si hay), el valor del primer parámetro después de la línea de formato se transforma y se saca conforme a la especificación establecida. La segunda especificación provoca la transformación y salida del segundo parámetro, y así sucesivamente hasta el fin de la línea de formato.

La especificación del formato tiene la siguiente forma:

%[flags][width][.precision][{h | l | ll | I32 | I64}]type

Cada campo de la especificación de formato es un símbolo simple o un número que denota una opción de formato corriente. La especificación de formato más simple contiene sólo el signo de porcentaje (%) y un símbolo que define el tipo de parámetro de salida (por ejemplo %s). Si en la línea de formato hay que mostrar el signo de porcentaje, es necesario usar la especificación de formato %%.

flags

© 2000-2025, MetaQuotes Ltd.

Bandera	Descripción	Comportamiento por defecto
- (menos)	Alineación por el lado izquierdo dentro del ancho establecido	Alineación por el lado derecho
+ (más)	Muestra de signos + o - para los tipos con signos	El signo se muestra sólo si el valor es negativo
0 (cero)	Antes de un valor de salida se añaden los ceros dentro del ancho establecido. Si la bandera **0** está especificada con un formato entero (**i**, **u**, **x**, **X**, **o**, **d**) y está determinada la especificación de precisión (por ejemplo, %04.d), entonces 0 se ignora.	Nada se añade
espacio	Antes de un valor de salida se pone un espacio, si el valor es de signo o positivo	Los espacios no se insertan
#	Si se usa junto con el formato **o**, **x** o **X**, entonces antes del valor de salida se añade 0, 0x o 0X respectivamente.	Nada se añade
	Si se usa junto con el formato **e**, **E**, **a** o **A**, el valor siempre se muestra con punto decimal.	El punto decimal se muestra sólo si hay una parte fraccionaria no nula
	Si se usa junto con el formato **g** o **G**, la bandera determina la presencia del punto decimal en el valor de salida e impide el recorte de ceros principales. La bandera # se ignora durante el uso compartido con los formatos **c**, **d**, **i**, **u**, **s**.	El punto decimal se muestra sólo si hay una parte fraccionaria no nula. Los ceros principales se cortan

width

El número decimal no negativo que establece el número mínimo de símbolos de salida del valor formateado. Si el número de símbolos de salida es menos que el ancho especificado, entonces se añade la cantidad correspondiente de espacios a la izquierda o a la derecha dependiendo de la alineación (bandera -). Si hay bandera cero (0), se añade la cantidad correspondiente de ceros antes del valor de salida. Si el número de símbolos de salida es más que el ancho especificado, entonces el valor de salida nunca se corta.

Si el asterisco (*) está especificado como el ancho, el valor del tipo int tiene que estar en la lista de parámetros pasados en el lugar correspondiente. Este valor va a ser usado para especificar el ancho del valor de salida.

precision

El número decimal no negativo que determina la precisión de salida, es decir, el número de cifras después del punto decimal. A diferencia de la especificación del ancho, la especificación de precisión puede recortar parte del valor fraccionario con redondeo o sin él.

© 2000-2025, MetaQuotes Ltd.

Para diferentes tipos (**type**) de formato la especificación de precisión se aplica de diferentes maneras.

Tipos	descripción	Comportamiento por defecto
a, A	La especificación de precisión fija el número de dígitos después del punto decimal.	Precisión por defecto - 6.
c, C	No se usa.	
d, i, u, o, x, X	Fija el número mínimo de cifras de salida. Si el número de cifras en el parámetro correspondiente es menos de la precisión indicada, el valor de salida se completa con ceros por la izquierda. El valor de salida no se recorta, si el número de cifras de salida es más de la precisión indicada.	Precisión por defecto - 1.
e, E, f	Fija el número de dígitos después del punto decimal. La última cifra se redondea.	Precisión por defecto - 6. Si está especificada la precisión 0 o falta la parte fraccionaria, el punto decimal no se muestra.
g, G	Fija el número máximo de cifras significantes.	Se muestran 6 cifras significantes.
s	Fija el número de símbolos de salida de una línea. Si el largo de una línea supera el valor de la precisión, esta línea se recorta en la salida.	Se muestra toda la línea.

```
PrintFormat("1. %s", _Symbol);
PrintFormat("2. %.3s", _Symbol);
int length=4;
PrintFormat("3. %.*s", length, _Symbol);
/*
1. EURUSD
2. EUR
3. EURU
/
```

h | l | ll | I32 | I64

Especificación de tamaños de datos pasados como parámetros.

Tipo de parámetro	Prefijo utilizado	Especificador compartido del tipo
int	l (L minúscula)	d, i, o, x, or X

© 2000-2025, MetaQuotes Ltd.

Tipo de parámetro	Prefijo utilizado	Especificador compartido del tipo
uint	l (L minúscula)	o, u, x, or X
long	ll (dos L minúsculas)	d, i, o, x, or X
short	h	d, i, o, x, or X
ushort	h	o, u, x, or X
int	I32	d, i, o, x, or X
uint	I32	o, u, x, or X
long	I64	d, i, o, x, or X
ulong	I64	o, u, x, or X

type

El especificador del tipo es el único campo obligatorio para la salida formateada.

Símbolo	Tipo	Formato de salida
c	int	Símbolo del tipo short (Unicode)
C	int	Símbolo del tipo char (ANSI)
d	int	Entero decimal con signo
i	int	Entero decimal con signo
o	int	Entero octal sin signo
u	int	Entero decimal sin signo
x	int	Entero hexadecimal sin signo, utilizando "abcdef"
X	int	Entero hexadecimal sin signo, utilizando "ABCDEF"
e	double	Valor real en formato [-]d.dddd **e** [sign]ddd, donde la d es una cifra decimal, dddd - una o más cifras decimales, ddd - número de tres cifras que determina el tamaño de exponente, sign - signo más o menos
E	double	Similar al formato e, salvo que el signo de exponente se imprime con mayúscula (**E** en vez de e)
f	double	Valor real en formato [-]dddd.dddd, donde dddd - una o más cifras decimales. El número de dígitos antes del punto decimal depende de la magnitud del valor del número. El número de dígitos después del punto decimal depende la precisión necesaria.

© 2000-2025, MetaQuotes Ltd.

Símbolo	Tipo	Formato de salida
g	double	Valor real mostrado en el formato **f** o **e**, dependiendo de cuál de las salidas va a ser más compacta.
G	double	Valor real mostrado en el formato **f** o **E**, dependiendo de cuál de las salidas va a ser más compacta.
a	double	Valor real en el formato [-]0xh.hhhh **p**±dd, donde h.hhhh es la mantisa en forma de cifras hexadecimales, usando "abcdef", dd - una o más cifras de la exponente. El número de dígitos después del punto decimal se determina por laespecificación de precisión
A	double	Valor real en el formato [-]0xh.hhhh **P**±dd, donde h.hhhh es la mantisa en forma de cifras hexadecimales, usando "ABCDEF", dd - una o más cifras de la exponente. El número de dígitos después del punto decimal se determina por la especificación de precisión
s	string	Muestra de línea

En vez de la función PrintFormat() se puede usar la función printf().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- nombre del servidor comercial
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- número de la cuenta comercial
   int login=(int)AccountInfoInteger(ACCOUNT_LOGIN);
//--- mostrar valor long
   long leverage=AccountInfoInteger(ACCOUNT_LEVERAGE);
   PrintFormat("%s %d: apalancamiento = 1:%I64d",
               server,login,leverage);
//--- divisa del depósito
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- mostrar valor double con 2 dígitos tras punto decimal
   double equity=AccountInfoDouble(ACCOUNT_EQUITY);
   PrintFormat("%s %d: tamaño de fondos propios en la cuenta = %.2f %s",
               server,login,equity,currency);
//--- mostrar valor double visualizando obligatoriamente el signo +/-
   double profit=AccountInfoDouble(ACCOUNT_PROFIT);
   PrintFormat("%s %d: resultado actual de posiciones abiertas = %+.2f %s",
               server,login,profit,currency);
//--- mostrar valor double con el número aleatorio de dígitos tras el punto decimal
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   string format_string=StringFormat("%%s: valor de un punto  = %%.%df",_Digits);
   PrintFormat(format_string,_Symbol,point_value);
//--- mostrar valor int
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   PrintFormat("%s: spread actual en puntos = %d ",
               _Symbol,spread);
//--- mostrar valor double en formato científico con el punto flotante y precisión de
   PrintFormat("DBL_MAX = %.17e",DBL_MAX);
//--- mostrar valor double en formato científico con el punto flotante y precisión de
   PrintFormat("EMPTY_VALUE = %.17e",EMPTY_VALUE);
//--- visualización a través PrintFormat() con precisión predefinida
   PrintFormat("PrintFormat(EMPTY_VALUE) = %e",EMPTY_VALUE);
//--- visualización simple a través de Print()
   Print("Print(EMPTY_VALUE) = ",EMPTY_VALUE);
/* resultado de ejecución
   MetaQuotes-Demo 1889998: apalancamiento = 1:100
   MetaQuotes-Demo 1889998: tamaño de fondos propios en la cuenta = 22139.86 USD
   MetaQuotes-Demo 1889998: resultado actual de posiciones abiertas = +174.00 USD
   EURUSD: valor de un punto  = 0.00001
   EURUSD: spread actual en puntos = 12
   DBL_MAX = 1.79769313486231570e+308
   EMPTY_VALUE = 1.79769313486231570e+308
   PrintFormat(EMPTY_VALUE) = 1.797693e+308
   Print(EMPTY_VALUE) = 1.797693134862316e+308
*/
  }
```

Véase también

StringFormat, DoubleToString, Tipos reales (double, float)

© 2000-2025, MetaQuotes Ltd.

ResetLastError

Pone el valor de la variable predefinida _LastError a cero.

```
void  ResetLastError();
```

Valor devuelto

No hay valor devuelto.

Nota

Cabe mencionar que la función GetLastError() no pone a cero la variable _LastError. Habitualmente la función ResetLastError() se invoca antes de llamar a la función después de la cual se comprueba la aparición de un error.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- reseteamos el código del último error antes de llamar la función
//--- dado que, en caso contrario, GetLastError() puede retornar el código del error a
   long lres=SymbolInfoInteger("123456",SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
   lres=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
   ResetLastError();
   lres=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

ResourceCreate

Esta función crea un recurso de imagen a base de un conjunto de datos. Hay dos variantes de esta función:

Crear recurso a base de un archivo

```
bool  ResourceCreate(
   const string      resource_name,       // nombre del recurso
   const string      path                 // ruta relativa hacia el archivo
   );
```

Crear archivo a base de un array de píxeles

```
bool  ResourceCreate(
   const string      resource_name,       // nombre del recurso
   const uint&       data[],              // conjunto de datos en forma del array
   uint              img_width,           // ancho del recurso de imagen a crear
   uint              img_height,          // alto del recurso de imagen a crear
   uint              data_xoffset,        // desplazamiento horizontal a la derecha de
   uint              data_yoffset,        // desplazamiento vertical hacia abajo de la
   uint              data_width,          // ancho total de la imagen basado en el cor
   ENUM_COLOR_FORMAT color_format         // modo de procesar el color
   );
```

Parámetros

resource_name

[in] Nombre del recurso.

data[][]

[in] Un array unidimensional o bidimencional (matriz) para crear una imagen completa.

img_width

[in] Ancho del área rectangular de la imagen en píxeles para colocarse en el recurso en forma de la imagen. No puede ser mayor que el valor de *data_width*.

img_height

[in] Alto del área rectangular de la imagen en píxeles para colocarse en el recurso en forma de la imagen.

data_xoffset

[in] Desplazamiento horizontal a la derecha del área rectangular de la imagen en píxeles.

data_yoffset

[in] Desplazamiento vertical hacia abajo del área rectangular de la imagen en píxeles.

data_width

[in] Se requiere sólo para los arrays unidimensionales, y significa el ancho total de la imagen que se crea a base del conjunto de datos. Si *data_width*=0, entonces se supone que es igual a *img_width*. Para los arrays bidimensionales este parámetro se ignora y se acepta que equivale a la segunda dimensión del array *data[]*.

color_format

[in] Modo de procesar el color desde la enumeración ENUM_COLOR_FORMAT.

© 2000-2025, MetaQuotes Ltd.

Valor devuelto

true – en caso del éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError(). Posibles errores:

- 4015 – ERR_RESOURCE_NAME_DUPLICATED (coincidencia de los nombres del recurso dinámico y estático),
- 4016 – ERR_RESOURCE_NOT_FOUND (recurso no encontrado),
- 4017 – ERR_RESOURCE_UNSUPPORTED_TYPE (tipo del recurso no se soporta),
- 4018 – ERR_RESOURCE_NAME_IS_TOO_LONG (nombre del recurso demasiado largo).

Nota

Si la segunda versión de la función se llama para la creación del mismo recurso con diferentes parámetros del ancho, alto y desplazamiento, el nuevo recurso no se vuelve a crear sino se actualiza el ya existente.

La primera variante de la función permite cargar las imágenes y sonidos desde archivos, mientras que la segunda versión sirve únicamente para la creación dinámica las imágenes.

Las imágenes deben ir en el formato BMP con la profundidad del color de 24 o 32 bits, los sonidos pueden tener sólo el formato WAV. El tamaño del recurso no puede ser más de 16 Mb.

ENUM_COLOR_FORMAT

Identificador	Descripción
COLOR_FORMAT_XRGB_NOALPHA	El componente del canal alfa se ignora
COLOR_FORMAT_ARGB_RAW	Los componentes del color no se procesan por el terminal (deben ser fijados correctamente por el usuario)
COLOR_FORMAT_ARGB_NORMALIZE	Los componentes del color se procesan por el terminal

Ejemplo:

```
//+------------------------------------------------------------------+
//| Actualiza los datos de los recursos gráficos                     |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- si se ha transmitido un tamaño cero, salimos
   if(width==0 || height==0)
      return;
//--- actualizamos los datos de los recursos y redibujamos el gráfico
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALI
      ChartRedraw();
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos los parámetros del recurso gráfico
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- establecemos el tamaño del array de píxeles
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- rellenamos la matriz de píxeles con un color transparente y creamos un recurso g
   ArrayInitialize(rc_data,0x00FFFFFF);
   if(!ResourceCreate(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMA
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- comprobamos el recurso gráfico creado.
//--- obtenemos los datos de hora y precio de la siguiente barra
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- creamos un objeto "Dibujo" en las coordenadas del precio y la hora del último ti
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- establecemos una anchura y una altura del objeto de dibujo creado igual a la anc
//--- establecemos el punto de anclaje del objeto en su centro.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- establecemos un recurso gráfico creado previamente como archivo de imagen para e
//--- en este caso, para especificar el nombre del recurso gráfico que se va a utiliza
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);

//--- establecemos el color DodgerBlue con una transparencia 200
   uint clr=ColorToARGB(clrDodgerBlue,200);
//--- rellenamos todo el array de píxeles del recurso gráfico con el color establecid
   ArrayInitialize(rc_data,clr);
//--- actualizamos los datos del recurso gráfico
   Update(rc_name,rc_data,rc_width,rc_height,true);
  }
```

Véase también

Recursos, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceFree

Elimina el recurso creado dinámicamente (libera la memoria ocupada por el recurso)

```
bool  ResourceFree(
   const string  resource_name      // nombre del recurso
   );
```

Parámetros

resource_name

[in] El nombre del recurso debe empezarse con "::".

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

La función ResourceFree() permite al desarrollador del programa mql5 administrar el consumo de la memoria en caso de trabajar con los recursos de forma activa. Los objetos gráficos vinculados al recurso que se elimina de la memoria van a mostrarse correctamente incluso después de su eliminación. Pero los objetos gráficos creados nuevamente (OBJ_BITMAP y OBJ_BITMAP_LABEL) ya no podrán utilizar el recurso eliminado.

La función elimina sólo los recursos dinámicos creados por este programa.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos los parámetros del recurso gráfico
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- establecemos el tamaño del array en píxeles
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- rellenamos la matriz de píxeles con un color transparente y creamos un recurso g
   ArrayInitialize(rc_data,0x00FFFFFF);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ResourceCreate(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMA
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- comprobamos el recurso gráfico creado.
//--- obtenemos los datos de hora y precio de la siguiente barra
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- creamos un objeto "Dibujo" en las coordenadas del precio y la hora del último ti
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- establecemos una anchura y una altura del objeto de dibujo creado igual a la anc
//--- establecemos el punto de anclaje del objeto en su centro.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- establecemos un recurso gráfico creado previamente como archivo de imagen para e
//--- en este caso, para especificar el nombre del recurso gráfico que se va a utiliza
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);

//--- establecemos el color DodgerBlue con una transparencia 200
   uint clr=ColorToARGB(clrDodgerBlue,200);
//--- rellenamos todo el array de píxeles del recurso gráfico con el color establecido
   ArrayInitialize(rc_data,clr);
//--- actualizamos los datos del recurso gráfico
   Update(rc_name,rc_data,rc_width,rc_height,true);

//--- esperamos tres segundos y cambiamos el color de la imagen
   Print("Wait 3 seconds before changing color");
   Sleep(3000);
//--- establecemos el color OrangeRed con una transparencia
   Print("Change color");
   clr=ColorToARGB(clrOrangeRed,200);
//--- rellenamos todo el array de píxeles del recurso gráfico con el color establecido
   ArrayInitialize(rc_data,clr);
//--- actualizamos los datos del recurso gráfico
   Update(rc_name,rc_data,rc_width,rc_height,true);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- esperamos tres segundos y liberamos el recurso gráfico
   Print("Wait 3 seconds before ResourceFree()");
   Sleep(3000);
   bool res=ResourceFree("::"+rc_name);
   Print("ResourceFree: ",res);

//--- intentamos cambiar el color después de liberar el recurso
   Print("Trying to change color to GreenYellow after ResourceFree()");
//--- establecemos el color GreenYellow con una transparencia 200
   clr=ColorToARGB(clrGreenYellow,200);
//--- rellenamos todo el array de píxeles del recurso gráfico con el color establecid
   ArrayInitialize(rc_data,clr);
//--- actualizamos los datos del recurso gráfico (el dibujo se mantiene, pero no se pu
   Update(rc_name,rc_data,rc_width,rc_height,true);
   Print("The color has not changed because the resource has been released");

//--- esperamos tres segundos y eliminamos el objeto de dibujo
   Print("Wait 3 seconds before deleting the Bitmap object");
   Sleep(3000);
   Print("Delete Bitmap object");
   ObjectDelete(0,obj_name);
  }
//+------------------------------------------------------------------+
//| Actualiza los datos del recurso gráfico                          |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- si se ha transmitido un tamaño cero, salimos
   if(width==0 || height==0)
      return;
//--- actualizamos los datos de los recursos y redibujamos el gráfico
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALI
      ChartRedraw();
  }
```

Véase también

Recursos, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceReadImage

Lee los datos del recurso gráfico creado con la función ResourceCreate() o guardado en el archivo EX5 tras la compilación.

```
bool  ResourceReadImage(
   const string      resource_name,       // nombre del recurso gráfico para leer
   uint&             data[],              // array para recibir datos desde el recurso
   uint&             width,               // para recibir el ancho de la imagen en el
   uint&             height,              // para recibir el alto de la imagen en el
   );
```

Parámetros

resource_name

[in] Nombre del recurso gráfico que contiene una imagen. Para acceder a sus propios recursos, el nombre se indica de forma corta "::resourcename". Si es necesario cargar un recurso desde el archivo EX5 compilado, hace falta el nombre completo indicando la ruta completa respecto a la carpeta MQL5, el nombre del archivo y el nombre del recurso - "path\\filename.ex5::resourcename".

data[][]

[in] Array unidimensional o bidimensional para recibir los datos desde el recurso gráfico.

img_width

[out] Ancho de la imagen del recurso gráfico en píxeles .

img_height

[out] Alto de la imagen del recurso gráfico en píxeles .

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Si luego a base del array *data[]* hay que crear un recurso gráfico, es necesario usar el formato del color COLOR_FORMAT_ARGB_NORMALIZE o COLOR_FORMAT_XRGB_NOALPHA.

Si el array *data[]* es bidimensional y su segunda dimensión es inferior al tamaño X(width) del recurso gráfico, la función ResourceReadImage() devolverá false y la lectura no será realizada. Pero en este caso si el recurso existe, en los parámetros width y height serán devueltos los tamaños actuales de la imagen. Esto permitirá hacer otro intento de recibir los datos desde el recurso.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- declaramos los parámetros del recurso gráfico
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- establecemos el tamaño del array en píxeles
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- rellenamos la matriz de píxeles con un color transparente y creamos un recurso g
   ArrayInitialize(rc_data,0x00FFFFFF);
   if(!ResourceCreate(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMA
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- comprobamos el recurso gráfico creado.
//--- obtenemos los datos de hora y precio de la siguiente barra
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- creamos un objeto "Dibujo" en las coordenadas del precio y la hora del último ti
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- establecemos una anchura y una altura del objeto de dibujo creado igual a la anc
//--- establecemos el punto de anclaje del objeto en su centro.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- establecemos un recurso gráfico creado previamente como archivo de imagen para e
//--- en este caso, para especificar el nombre del recurso gráfico que se va a utiliza
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);

//--- establecemos el color DodgerBlue con una trasparencia 200
   uint clr=ColorToARGB(clrDodgerBlue,200);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- rellenamos todo el array de píxeles del recurso gráfico con el color establecido
   ArrayInitialize(rc_data,clr);
//--- actualizamos los datos del recurso gráfico
   Update(rc_name,rc_data,rc_width,rc_height,true);

//--- esperamos 3 segundos antes de leer los datos del recurso gráfico
   Print("Wait 3 seconds before ResourceReadImage()");
   Sleep(3000);
//--- leemos la imagen del recurso en un nuevo array de píxeles
   uint rc_data_copy[];
   uint w=0,h=0;
   ResetLastError();
   if(!ResourceReadImage("::"+rc_name,rc_data_copy,w,h))
     {
      Print("ResourceReadImage() failed. Error code: ",GetLastError());
      return;
     }

//--- establecemos el color OrangeRed con una transparencia
   clr=ColorToARGB(clrOrangeRed,200);
//--- rellenamos todo el array de píxeles con el color establecido y creamos un nuevo
   ArrayInitialize(rc_data_copy,clr);
   if(!ResourceCreate(rc_name+"Copy",rc_data_copy,rc_width,rc_height,0,0,0,COLOR_FORMA
     {
      Print("New ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created new recource array: ",rc_data_copy.Size());

//--- creamos un objeto "Etiqueta gráfica" en las coordenadas del precio y la hora del
   string obj_name2="BitmapLabel";
   if(!ObjectCreate(0,obj_name2,OBJ_BITMAP_LABEL,0,0,0))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- obtenemos las coordenadas de pantalla según el precio y la hora obtenidos previa
   int x=0,y=0;
   if(!ChartTimePriceToXY(0,0,tick.time,tick.bid,x,y))
     {
      Print("New ChartTimePriceToXY() failed. Error code: ",GetLastError());
      return;
     }
//--- establecemos una anchura y una altura del objeto de etiqueta gráfica creado igua
//--- establecemos el punto de anclaje del objeto en su centro.
   ObjectSetInteger(0,obj_name2,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name2,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name2,OBJPROP_ANCHOR,ANCHOR_LEFT_UPPER);
   ObjectSetInteger(0,obj_name2,OBJPROP_XDISTANCE,x);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(0,obj_name2,OBJPROP_YDISTANCE,y);
//--- establecemos el recurso gráfico copiado como archivo de imagen para el objeto de
//--- en este caso, para especificar el nombre del recurso gráfico que se va a utiliza
   ObjectSetString(0,obj_name2,OBJPROP_BMPFILE,"::"+rc_name+"Copy");

//--- cambiamos el color de un nuevo objeto de etiqueta gráfica
   Print("Wait 3 seconds before changing color to GreenYellow");
   Sleep(3000);
//--- establecemos el color GreenYellow con una transparencia 200
   clr=ColorToARGB(clrGreenYellow,200);
//--- rellenamos todo el array de píxeles del recurso gráfico con el color establecido
   ArrayInitialize(rc_data_copy,clr);
//--- actualizamos los datos del recurso gráfico
   Update(rc_name+"Copy",rc_data_copy,rc_width,rc_height,true);

//--- esperamos tres segundos y eliminamos los recursos y ambos objetos
   Print("Wait 3 seconds before deleting both objects");
   Sleep(3000);
   Print("Deleting Resource and all Bitmap objects");
   ResourceFree("::"+rc_name);
   ResourceFree("::"+rc_name+"Copy");
   ObjectDelete(0,obj_name);
   ObjectDelete(0,obj_name2);
  }
//+------------------------------------------------------------------+
//| Actualiza los datos del recurso gráfico                          |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- si se ha transmitido un tamaño cero, salimos
   if(width==0 || height==0)
      return;
//--- actualizamos los datos de los recursos y redibujamos el gráfico
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZ
      ChartRedraw();
  }
```

Véase también

Recursos, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceSave

Guarda el recurso en el archivo especificado.

```
bool  ResourceSave(
   const string  resource_name      // nombre del recurso
   const string  file_name          // nombre del archivo
   );
```

Parámetros

resource_name

[in] El nombre del recurso debe empezarse con "::".

file_name

[in] Nombre del archivo respecto a MQL5\Files.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

La función siempre sobrescribe el archivo, creando si precisa todas las subcarpetas intermedias en el nombre del archivo, en caso de no haberlas.

Véase también

Recursos, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

Ejemplo:

```
//--- parámetros del recurso gráfico
string ExtResName="Resource";
int    ExtResWidth=100;
int    ExtResHeight=100;
uint   ExtResData[];
int    ExtResSize=ExtResWidth*ExtResHeight;

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   ResetLastError();
//--- establecemos el tamaño del array en píxeles
   if(ArrayResize(ExtResData,ExtResSize)!=ExtResSize)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- rellenamos la matriz de píxeles con un color transparente y creamos un recurso g
   ArrayInitialize(ExtResData,0x00FFFFFF);
   if(!ResourceCreate(ExtResName,ExtResData,ExtResWidth,ExtResHeight,0,0,0,COLOR_FORMA
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",ExtResData.Size());

//--- comprobamos el recurso gráfico creado.
//--- obtenemos los datos de hora y precio de la siguiente barra
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- creamos un objeto "Dibujo" en las coordenadas del precio y la hora del último ti
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- establecemos una anchura y una altura del objeto de dibujo creado igual a la anc
//--- establecemos el punto de anclaje del objeto en su centro.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,ExtResWidth);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,ExtResHeight);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- establecemos un recurso gráfico creado previamente como archivo de imagen para e
//--- en este caso, para especificar el nombre del recurso gráfico que se va a utiliza
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+ExtResName);

//--- establecemos el color GreenYellow con una transparencia 200
   uint clr=ColorToARGB(clrGreenYellow,200);
//--- rellenamos todo el array de píxeles del recurso gráfico con el color establecido
   ArrayInitialize(ExtResData,clr);
//--- dibujamos una cuadrícula con el color DodgerBlue
   for(int x=0;x<ExtResWidth;x+=9)
      LineVertical(x,0,ExtResHeight,ColorToARGB(clrDodgerBlue,200));
   for(int y=0;y<ExtResHeight;y+=9)
      LineHorizontal(0,ExtResWidth,y,ColorToARGB(clrDodgerBlue,200));
//--- actualizamos los datos del recurso gráfico
   Update(ExtResName,ExtResData,ExtResWidth,ExtResHeight,true);

//--- esperamos 3 segundos antes de eliminar el recurso y el objeto gráfico
   Print("Wait 3 seconds before deleting the resource and graphic object");
   Sleep(3000);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- guardamos el recurso en un archivo
   ResetLastError();
   if(!ResourceSave("::"+ExtResName,"ResourceSave\\"+ExtResName+".bmp"))
     {
      Print("ResourceSave() failed. Error code: ",GetLastError());
      return;
     }
//--- eliminamos el recurso y el objeto de dibujo gráfico
   if(!ResourceFree("::"+ExtResName))
     {
      Print("ResourceFree() failed. Error code: ",GetLastError());
      return;
     }
   if(!ObjectDelete(0,obj_name))
     {
      Print("ObjectDelete() failed. Error code: ",GetLastError());
      return;
     }

//--- ahora creamos un nuevo objeto "Etiqueta gráfica" en las coordenadas del precio y
//--- obtenemos las coordenadas de pantalla según el precio y la hora obtenidos previa
   int x=0,y=0;
   if(!ChartTimePriceToXY(0,0,tick.time,tick.bid,x,y))
     {
      Print("ChartTimePriceToXY() failed. Error code: ",GetLastError());
      return;
     }
   obj_name="BitmapLabel";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP_LABEL,0,0,0))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- establecemos el punto de anclaje del objeto en su centro.
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- establecemos para el objeto de etiqueta gráfica las coordenadas del precio y la
   ObjectSetInteger(0,obj_name,OBJPROP_XDISTANCE,x-ExtResWidth);
   ObjectSetInteger(0,obj_name,OBJPROP_YDISTANCE,y);
//--- especificamos el recurso previamente guardado como una imagen bmp para el objeto
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"\\Files\\ResourceSave\\"+ExtResName+".b
//--- actualizamos el gráfico
   ChartRedraw();

//--- esperamos 3 segundos antes de eliminar el objeto de etiqueta gráfica
   Print("Wait 3 seconds before deleting the new graphic object");
   Sleep(3000);
   if(!ObjectDelete(0,obj_name))
      Print("ObjectDelete() failed. Error code: ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Actualiza los datos de los recursos gráficos                     |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- si se ha transmitido un tamaño cero, salimos
   if(width==0 || height==0)
      return;
//--- actualizamos los datos de los recursos y redibujamos el gráfico
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZ
      ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Dibuja una línea vertical                                        |
//+------------------------------------------------------------------+
void LineVertical(int x,int y1,int y2,const uint clr)
  {
   int tmp;
//--- clasificación según Y
   if(y1>y2)
     {
      tmp=y1;
      y1 =y2;
      y2 =tmp;
     }
//--- línea furea de la imagen
   if(y2<0 || y1>=ExtResHeight || x<0 || x>=ExtResWidth)
      return;
//--- permanecemos dentro de los límites de la imagen
   if(y1<0)
      y1=0;
   if(y2>=ExtResHeight)
      y2=ExtResHeight-1;
//--- dibujamos la línea
   int index=y1*ExtResWidth+x;
   for(int i=y1; i<=y2; i++,index+=ExtResWidth)
      ExtResData[index]=clr;
  }
//+------------------------------------------------------------------+
//| Dibuja una línea horizontal                                      |
//+------------------------------------------------------------------+
void LineHorizontal(int x1,int x2,int y,const uint clr)
  {
   int tmp;
//--- clasificación según X
   if(x1>x2)
     {
      tmp=x1;
      x1 =x2;
```

© 2000-2025, MetaQuotes Ltd.

```
      x2 =tmp;
     }
//--- línea furea de la imagen
   if(x2<0 || x1>=ExtResWidth || y<0 || y>=ExtResHeight)
      return;
//--- permanecemos dentro de los límites de la imagen
   if(x1<0)
      x1=0;
   if(x2>=ExtResWidth)
      x2=ExtResWidth-1;
//--- dibujamos la línea
   ArrayFill(ExtResData,y*ExtResWidth+x1,(x2-x1)+1,clr);
  }
```

© 2000-2025, MetaQuotes Ltd.

SetReturnError

Establece el código que retornará el proceso del terminal al completar el trabajo.

```
void  SetReturnError(
   int ret_code      // código de finalización del terminal de cliente
   );
```

Parámetros

ret_code

[in] El código de retorno devuelto por el proceso del terminal de cliente al completar el trabajo.

Valor retornado

No hay valor retornado.

Observación

La instalación del código de retorno *ret_code* utilizando la función SetReturnError() resultará útil para analizar las causas de la finalización del programa al iniciar el terminal desde la línea de comandos.

La función SetReturnError(), a diferencia de TerminalClose(), no finaliza el funcionamiento del terminal, solo define el código de retorno que devolverá el proceso del terminal al finalizarse.

Si la función SetReturnError() ha sido llamada varias veces y/o desde diferentes programas MQL5, el terminal devolverá el último código de retorno establecido.

El código establecido se retornará al finalizar el proceso del terminal, excepto en los siguientes casos:

- si ha tenido lugar un error crítico durante la ejecución;
- si se ha llamado la función TerminalClose(int ret_code), que da al terminal un comando para finalizar el trabajo con un código indicado.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   matrix matrix_a =
     {
        {-3.474589, 1.106384, -9.091977,-3.925227 },
        {-5.522139, 2.366887,-15.162351,-6.357512 },
        { 8.394926,-2.960067, 22.292115, 9.524129 },
        { 7.803242,-2.080287, 19.217706, 8.186645 }
     };
   matrix matrix_l(4,4);
   matrix matrix_u(4,4);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- decomposición LU
   matrix_a.LU(matrix_l,matrix_u);

//--- comprobamos que A = L * U sea correcto
   matrix matrix_lu=matrix_l.MatMul(matrix_u);
   int    compare_errors=(int)matrix_a.Compare(matrix_lu,1e-29);
   Print("MatrixCompare errors=",compare_errors);

//--- a su finalización, el terminal de cliente retornará el número de errores de comp
   SetReturnError(compare_errors);
  }
```

Ver también

Ejecución de programas, Errores de ejecución, Causas de desinicialización, TerminalClose

© 2000-2025, MetaQuotes Ltd.

SetUserError

Pone la variable predefinida _LastError en el valor equivalente a ERR_USER_ERROR_FIRST + user_error

```
void  SetUserError(
   ushort user_error,    // número del error
   );
```

Parámetros

user_error

[in] Número del error fijado por el usuario.

Valor devuelto

No hay valor devuelto.

Nota

Después de haber fijado el error con la función SetUserError(user_error), La función GetLastError() devolverá el valor equivalente a ERR_USER_ERROR_FIRST + user_error.

Ejemplo:

```
void OnStart()
  {
//--- fijamos el número del error 65537=(ERR_USER_ERROR_FIRST +1)
   SetUserError(1);
//--- obtendremos el código del último error
   Print("GetLastError = ",GetLastError());
/*
   Resultado
   GetLastError = 65537
*/
  }
```

© 2000-2025, MetaQuotes Ltd.

Sleep

La función suspende la ejecución del Asesor Experto en curso o del script por un intervalo determinado.

```
void  Sleep(
   int  milliseconds      // intervalo
   );
```

Parámetros

milliseconds

[in] Intervalo de retraso en milisegundos.

Valor devuelto

No hay valor devuelto.

Nota

La función Sleep() no puede ser llamada de los indicadores personalizados, porque los indicadores se ejecutan en un hilo de interfaz y no deben frenarlo. La función dispone de la comprobación "built-in" del estado de bandera de interrupción de Asesor Experto cada 0.1 segundos.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- mostramos en el comentario del gráfico una cuenta atrás de 10 a 1
   for(int i=10; i>0 && !IsStopped(); i--)
     {
      Comment(StringFormat("Wait %u seconds",i));
      Sleep(1000);
     }
//--- escribimos en el comentario "entrante" un texto describiendo el propósito del sc
   string text="This was a test showing how the Sleep() function works";
   string mess="";
   for(int i=0; i<(int)text.Length(); i++)
     {
      mess+=ShortToString(text.GetChar(i));
      Sleep(100);
      Comment(mess);
     }
//--- nos despedimos...
   Sleep(1000);
   for(int i=0; i<6; i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      mess=(i % 2 == 0 ? "" : "  Bye!");
      Comment(mess);
      Sleep(300);
     }
//--- borramos el texto en el gráfico
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalClose

Envía al terminal el comando de finalizar el trabajo.

```
bool  TerminalClose(
   int ret_code      // código de cierre del terminal de cliente
   );
```

Parámetros

ret_code

[in] Código de devolución devuelto por el proceso del terminal de cliente al terminar su trabajo.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

La función TerminalClose() no detiene el terminal inmediatamente, sólo le manda un comando para completar su operatividad.

El código de Asesor Experto que ha llamado a TerminalClose() tiene que realizar todos los preparativos para la finalización inmediata del trabajo (por ejemplo, hay que cerrar todos los archivos abiertos previamente de una forma correcta). Después de la llamada a esta función tiene que seguir el operador return.

El parámetro *ret_code* permite indicar el código de devolución necesario para analizar las causas de finalización programada del funcionamiento del terminal cuando éste se inicia desde la línea de comandos.

Ejemplo:

```
//--- input parameters
input int  tiks_before=500; // número de ticks hasta la finalización
input int  pips_to_go=15;   // distancia en pips
input int  seconds_st=50;   // los segundos que damos al Asesor Experto
//--- globals
datetime   launch_time;
int        tick_counter=0;
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(__FUNCTION__," reason code = ",reason);
   Comment("");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   static double first_bid=0.0;
   MqlTick       tick;
   double        distance;
//---
   SymbolInfoTick(_Symbol,tick);
   tick_counter++;
   if(first_bid==0.0)
     {
      launch_time=tick.time;
      first_bid=tick.bid;
      Print("first_bid = ",first_bid);
      return;
     }
//--- distancia de precio en pips
   distance=(tick.bid-first_bid)/_Point;
//--- mostramos una notificación para monitorear el trabajo del AE
   string comm="Desde el momento de inicio:\r\n\x25CF han pasado segundos: "+
               IntegerToString(tick.time-launch_time)+" ;"+
               "\r\n\x25CF ticks que se ha recibido: "+(string)tick_counter+" ;"+
               "\r\n\x25CF precio ha ido en puntos: "+StringFormat("%G",distance);
   Comment(comm);
//--- sección de comprobación de condiciones para el cierre del terminal
   if(tick_counter>=tiks_before)
      TerminalClose(0);    // salida por el contador de ticks
   if(distance>pips_to_go)
      TerminalClose(1);    // vamos arriba al número de pips igual a pips_to_go
   if(distance<-pips_to_go)
      TerminalClose(-1);   // vamos abajo al número de pips igual a pips_to_go
   if(tick.time-launch_time>seconds_st)
      TerminalClose(100);  // trabajo se termina por expiración de plazo
//---
  }
```

Véase también

Funcionamiento de programas, Errores de ejecución, Razones de deinicialización

© 2000-2025, MetaQuotes Ltd.

TesterHideIndicators

Establece el modo de muestra/ocultación de los indicadores que se usan en el experto. La funcion se ha diseñado para gestionar la visibilidad de los indicadores utilizados solo en la simulación.

```
void  TesterHideIndicators(
   bool      hide     // bandera
   );
```

Parámetros

hide

[in] Bandera de ocultación de indicadores en la simulación. Establezca true, si desea ocultar los indicadores creados, de lo contrario, false.

Valor devuelto

Ninguno.

Nota

Por defecto, en el gráfico de simulación visual se muestran todos los indicadores que se crean en el experto simulado. Asimismo, estos indicadores se muestran en el gráfico que se abre de forma automática al finalizar la simulación. La función TesterHideIndicators() permite al desarrollador escribir en el código del experto la prohibición de mostrar los indicadores utilizados.

Para prohibir la muestra del indicador utilizado al simular el asesor, es necesario, antes de crear su manejador, llamar TesterHideIndicators() con el parámetro true: todos los indicadores creados después de ello se marcarán con la bandera de ocultación. Los indicadores marcados con la bandera de ocultación tampoco se muestran en la simulación visual, en el gráfico que se abre automáticamente al finalizar la simulación.

Para desactivar el modo de ocultación de los indicadores creados nuevamente, debemos llamar de nuevo TesterHideIndicators(), pero con el parámetro false. En el gráfico de simulación se pueden mostrar solo aquellos indicadores que se creen directamente desde el experto simulado. Esta regla también se refiere a aquellos casos en los que no hay ni una sola plantilla en la carpeta <catálogo_de_datos>MQL5\Profiles\Templates.

Si en la carpeta <catálogo_de_datos>MQL5\Profiles\Templates está la plantilla especial <nombre_del_experto>.tpl, en el modo de simulación visual y en el gráfico de simulación se mostrarán solo los indicadores de esta plantilla. En este caso, no se mostrará ningún indicador usado en el experto simulado. Incluso si en el código del simulador se ha llamado la función TesterHideIndicators() con el parámetro true.

Si en la carpeta <catálogo_de_datos>MQL5\Profiles\Templates no existe la plantilla especial <nombre_del_experto>.tpl, pero existe la plantilla tester.tpl, en la simulación visual y en el gráfico de simulación se mostrarán los indicadores de la plantilla tester.tpl y los indicadores de la plantilla cuya muestra no haya sido prohibida con la ayuda de la función TesterHideIndicators(). Si no hay plantilla tester.tpl, en su lugar se usarán los indicadores de la plantilla default.tpl.

Si el simulador de estrategias no encuentra ni una plantilla conveniente (<nombre_del_experto>.tpl, tester.tpl o default.tpl), la muestra de los indicadores usados en el asesor será completamente controlada mediante la función TesterHideIndicators().

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
bool CSampleExpert::InitIndicators(void)
  {
   TesterHideIndicators(true);
//--- create MACD indicator
   if(m_handle_macd==INVALID_HANDLE)
      if((m_handle_macd=iMACD(NULL,0,12,26,9,PRICE_CLOSE))==INVALID_HANDLE)
        {
         printf("Error creating MACD indicator");
         return(false);
        }
   TesterHideIndicators(false);
//--- create EMA indicator and add it to collection
   if(m_handle_ema==INVALID_HANDLE)
      if((m_handle_ema=iMA(NULL,0,InpMATrendPeriod,0,MODE_EMA,PRICE_CLOSE))==INVALID_H
        {
         printf("Error creating EMA indicator");
         return(false);
        }
//--- succeed
   return(true);
  }
```

Vea también

IndicatorRelease

© 2000-2025, MetaQuotes Ltd.

TesterStatistics

La función devuelve el valor del indicador estadístico especificado que ha sido calculado a base de los resultados de simulación

```
double  TesterStatistics(
   ENUM_STATISTICS statistic_id      // identificador
   );
```

Parámetros

statistic_id

[in] El identificador del indicador estadístico desde la enumeración ENUM_STATISTICS.

Valor devuelto

El valor del indicador estadístico desde los resultados de simulación.

Nota

La función puede ser llamada dentro de OnTester() o OnDeinit() en el probador de estrategias. En otras ocasiones el resultado no está definido.

Ejemplo:

```
// El asesor basado en el archivo estándar "MACD Sample.mq5"
// muestra el resultado de la ejecución TesterStatistics() en el manejador del evento
#define MACD_MAGIC 1234502
//---
#include <Trade\Trade.mqh>
#include <Trade\SymbolInfo.mqh>
#include <Trade\PositionInfo.mqh>
#include <Trade\AccountInfo.mqh>
//---
input double InpLots          =0.1; // Lots
input int    InpTakeProfit    =50;  // Take Profit (in pips)
input int    InpTrailingStop  =30;  // Trailing Stop Level (in pips)
input int    InpMACDOpenLevel =3;   // MACD open level (in pips)
input int    InpMACDCloseLevel=2;   // MACD close level (in pips)
input int    InpMATrendPeriod =26;  // MA trend period
//---
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- creamos todos los objetos necesarios
   if(!ExtExpert.Init())
      return(INIT_FAILED);
//--- la inicialización ha tenido éxito
   return(INIT_SUCCEEDED);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert new tick handling function                                |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   static datetime limit_time=0; // última hora de la llamada considerando timeout
//--- si la hora supera el valor establecido limit_time
   if(TimeCurrent()>=limit_time)
     {
      //--- comprobamos los datos
      if(Bars(Symbol(),Period())>2*InpMATrendPeriod)
        {
         //--- en caso de ejecución exitosa, aumentamos limit_time en timeout segundos
         if(ExtExpert.Processing())
            limit_time=TimeCurrent()+ExtTimeOut;
        }
     }
  }
//+------------------------------------------------------------------+
//| Expert tester handling function                                  |
//+------------------------------------------------------------------+
double OnTester(void)
  {
   double ret=TesterStatistics(STAT_PROFIT_FACTOR);
   double profit=TesterStatistics(STAT_PROFIT);
   int    trades_total=(int)TesterStatistics(STAT_TRADES);
   int    profit_total=(int)TesterStatistics(STAT_PROFIT_TRADES);
   int    loss_total=(int)TesterStatistics(STAT_LOSS_TRADES);
   PrintFormat("%s: Profit = %.2f, trades total: %lu, profit trades total: %lu, loss t
   return(ret);
  /*
  Resultado:
   OnTester: Profit = 209.84, trades total: 13, profit trades total: 11, loss trades t
   final balance 10209.84 USD
   OnTester result 3.020606644198363
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

TesterStop

Da el comando para finalizar el trabajo del programa durante la simulación.

```
void  TesterStop();
```

Valor retornado

No hay valor retornado.

Observación

La función TesterStop() ha sido pensada para finalizar de forma rutinaria y anticipada el funcionamiento de un asesor en un agente de simulación, por ejemplo, al alcanzar el número de establecido de transacciones o el nivel de reducción indicado.

La llamada de TesterStop() se considera una finalización normal de la simulación, por eso se llamará la función OnTester(), proporcionando al simulador de estrategias todas las estadísticas comerciales acumuladas y los valores del criterio de optimización.

La llamada de ExpertRemove() en el simulador de estrategias también significa la finalización normal de la simulación y permite obtener estadísticas comerciales, pero además, en este caso, el asesor es descargado de la memoria del agente. En esta situación, para ejecutar una pasada en el siguiente conjunto de parámetros, se necesitará tiempo para cargar de nuevo el programa. Por eso, para finalizar de forma rutinaria y anticipada el funcionamiento de la simulación, el uso de TesterStop() es la opción preferible.

Ejemplo:

```
//--- defines
#define BALANCE_LOSS_STOP  100.0       // valor de reducción del balance al que se det
#define EQUITY_LOSS_STOP   100.0       // valor de reducción de los recursos al que se

//--- input parameters
input  double  InpLots        =  0.1;  // lotes
input  uint    InpStopLoss    =  50;   // Stop loss en puntos
input  uint    InpTakeProfit  =  150;  // Take Profit  en puntos
sinput ulong   InpMagic       =  123;  // MagicNumber
sinput ulong   InpDeviation   =  5;    // desviación
//--- global variables
CTrade      trade;                     // ejemplar de la clase comercial
CSymbolInfo symb;                      // ejemplar de la clase de símbolo
CAccountInfo account;                  // ejemplar de clase de cuenta comercial
...
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- la inicialización ha tenido éxito
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- actualizamos las cotizaciones actuales
   if(!symb.RefreshRates())
      return;
   ...

//--- si el balance o los fondos se han reducido más de lo indicado en las macrosusti
//--- consideramos que la prueba ha fallado y llamamos a la función TesterStop()
//--- comprobamos si la reducción de balance ha sido superior a BALANCE_LOSS_STOP
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()<balance_prev-BALANCE_LOSS_STOP)
        {
         PrintFormat("The initial balance of %.2f %s decreased by %.2f %s, and now has
         TesterStop();
         /*
        Resultado:
         The initial balance of 10000.00 USD decreased by 100.10 USD, and now has a va
         TesterStop() called on 9% of testing interval
         */
        }
     }
//--- comprobamos si la pérdida de recursos ha sido superior a EQUITY_LOSS_STOP
   if(equity_prev!=account.Equity())
     {
      if(account.Equity()<equity_prev-EQUITY_LOSS_STOP)
        {
         PrintFormat("The initial equity of %.2f %s decreased by %.2f %s, and now has
         TesterStop();
         /*
        Resultado:
         The initial equity of 10000.00 USD decreased by 100.10 USD, and now has a val
         TesterStop() called on 9% of testing interval
         */
        }
     }
  }
```

Ver también

Ejecución de programas, Simulación de estrategias comerciales, ExpertRemove, SetReturnError

© 2000-2025, MetaQuotes Ltd.

TesterDeposit

Función especial para emular operaciones de depósito de fondos durante la simulación. Se puede utilizar en algunos sistemas de gestión de capital.

```
bool  TesterDeposit(
   double money      // suma a depositar
   );
```

Parámetros

money

[in] Suma a ingresar en la cuenta en la divisa del depósito.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Ejemplo:

```
//--- defines
#define BALANCE_LOSS_DEPOSIT  100.0    // valor de reducción del balance a partir del

//--- input parameters
input  double  InpLots        =  0.1;  // Lots
input  uint    InpStopLoss    =  50;   // Stop loss in points
input  uint    InpTakeProfit  =  150;  // Take Profit in points
sinput ulong   InpMagic       =  123;  // Magic number
sinput ulong   InpDeviation   =  5;    // Deviation
//--- global variables
CTrade      trade;                     // ejemplar de la clase comercial
CSymbolInfo symb;                      // ejemplar de la clase de símbolo
CAccountInfo account;                  // ejemplar de clase de cuenta comercial
...
double      balance_dep_summ;          // suma total de las recargas de balance
uint        balance_dep_total;         // número de recargas de balance
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- guardamos los valores del balance inicial
   balance_prev=account.Balance();
   balance_dep_summ=0;
   balance_dep_total=0;
//--- la inicialización ha tenido éxito
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- actualizamos las cotizaciones actuales
   if(!symb.RefreshRates())
      return;
   ...

//--- si el balance se ha reducido más de lo establecido en la macrosustitución BALANC
//--- significa que consideramos imprescindible recargar la cuenta, y llamamos a la fu
//--- comprobamos si la pérdida de balance ha sido superior a BALANCE_LOSS_DEPOSIT
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()<balance_prev-BALANCE_LOSS_DEPOSIT)
        {
         double loss=balance_prev-account.Balance();
         PrintFormat("The initial balance of %.2f %s decreased by %.2f %s. It is neces
         if(TesterDeposit(loss))
           {
            balance_dep_total++;
            balance_dep_summ+=loss;
            balance_prev=account.Balance();
            PrintFormat("Funds have been deposited into the account. Account balance:
            PrintFormat("Total deposits: %lu. Amount of deposits: %.2f %s.",balance_de
           }
        /*
        Resultado:
         The initial balance of 10000.00 USD decreased by 116.00 USD. It is necessary
         deal #45 balance 116.00 [deposit] done
         Funds have been deposited into the account. Account balance: 10000.00 USD.
         Total deposits: 1. Amount of deposits: 116.00 USD.
        */
         }
     }
  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- el valor de salida del manejador será el valor de la reducción máxima de balance
   double ret=TesterStatistics(STAT_BALANCE_DD);
//--- mostramos un mensaje sobre la reducción de fondos, el número de recargas y su in
   PrintFormat("%s: Maximum balance drawdown in money: %.2f %s. Total deposits: %lu. A
//--- retornamos el resultado
   return(ret);
  /*
  Resultado:
```

© 2000-2025, MetaQuotes Ltd.

```
   OnTester: Maximum balance drawdown in money: 5188.50 USD. Total deposits: 46. Amour
   final balance 4867.50 USD
   OnTester result 5188.5
  */
  }
```

Vea también

TesterDeposit

© 2000-2025, MetaQuotes Ltd.

TesterWithdrawal

Es una función especial para emular las operaciones de retiro de fondos durante el proceso de evaluación. Puede usarse en algunos sistemas de administración del capital.

```
bool  TesterWithdrawal(
   double money      // importe a retirar
   );
```

Parámetros

money

[in] Cuantía del dinero que hay que retirar de la cuenta (en divisa del depósito).

Valor devuelto

En caso del éxito devuelve true, de lo contrario false.

Ejemplo:

```
//--- defines
#define BALANCE_PROFIT_WITHDRAWAL   5  // valor del beneficio en el balance al que se

//--- input parameters
input  double  InpLots        =  0.1;  // Lots
input  uint    InpStopLoss    =  50;   // Stop loss in points
input  uint    InpTakeProfit  =  150;  // Take Profit in points
sinput ulong   InpMagic       =  123;  // Magic number
sinput ulong   InpDeviation   =  5;    // Deviation
//--- global variables
CTrade      trade;                     // ejemplar de la clase comercial
CSymbolInfo symb;                      // ejemplar de la clase de símbolo
CAccountInfo account;                  // ejemplar de clase de cuenta comercial
...
double      balance_op_sum;            // suma total de las operaciones de balance
uint        balance_op_total;          // cantidad total de operaciones de balance
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- guardamos los valores del balance inicial
   balance_prev=account.Balance();
   balance_op_sum=0;
   balance_op_total=0;
//--- la inicialización ha tenido éxito
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- actualizamos las cotizaciones actuales
   if(!symb.RefreshRates())
      return;
   ...

//--- si el beneficio del balance es superior al balance actual en el valor especifica
//--- consideramos que es necesario retirar estos fondos de la cuenta y llamar a la fu
//--- comprobamos si el beneficio en el balance es superior a BALANCE_PROFIT_WITHDRAWA
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()>balance_prev+BALANCE_PROFIT_WITHDRAWAL)
        {
         double profit=account.Balance()-balance_prev;
         PrintFormat("The account balance has been increased by %.2f %s. Need to withd
         if(TesterWithdrawal(profit))
           {
            balance_op_total++;
            balance_op_summ+=profit;
            balance_prev=account.Balance();
            PrintFormat("Funds have been withdrawn from the account. Account balance:
            PrintFormat("Total withdrawals: %lu. Amount of withdrawals: %.2f %s.",bala
           }
        /*
        Resultado:
         The account balance has been increased by 21.00 USD. Need to withdraw these f
         deal #13 balance -21.00 [withdrawal] done
         Funds have been withdrawn from the account. Account balance: 10000.00 USD.
         Total withdrawals: 1. Amount of withdrawals: 21.00 USD.
        */
        }
     }
  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- el valor de salida del manejador será el valor de la reducción máxima de balance
   double ret=TesterStatistics(STAT_BALANCE_DD);
//--- registremos la reducción, el número de retiradas y su importe total
   PrintFormat("%s: Maximum balance drawdown in money: %.2f %s. Total withdrawals: %lu
//--- retornamos el resultado
   return(ret);
  /*
  Resultado:
```

© 2000-2025, MetaQuotes Ltd.

```
   OnTester: Maximum balance drawdown in money: 5188.50 USD. Total withdrawals: 2. Amo
   final balance 4867.50 USD
   OnTester result 5188.5
  */
  }
```

Vea también

TesterDeposit

© 2000-2025, MetaQuotes Ltd.

TranslateKey

Retorna un símbolo Unicode según el código virtual de una tecla, teniendo en cuenta el idioma de entrada y el estado de las teclas de control.

```
short  TranslateKey(
   int  key_code      // código de la tecla para obtener un símbolo Unicode
   );
```

Parámetros

key_code

[in] Código de la tecla.

Valor devuelto

Símbolo Unicode, en caso de que la transformación tenga éxito. En caso de error, la función retorna -1.

Nota

La función utiliza ToUnicodeEx para transformar las teclas pulsadas por el usuario en símbolos Unicode. Puede surgir un error en el caso de que ToUnicodeEx no se accione, por ejemplo, al intentar obtener un símbolo para la tecla SHIFT.

Ejemplo:

```
void OnChartEvent(const int id,const long& lparam,const double& dparam,const string& s
  {
   if(id==CHARTEVENT_KEYDOWN)
     {
      short sym=TranslateKey((int)lparam);
      //--- si el símbolo introducido se ha transformado con éxito en Unicode
      if(sym>0)
         Print(sym,"'",ShortToString(sym),"'");
      else
         Print("Error in TranslateKey for key=",lparam);
     }
  }
```

Vea también

Eventos del terminal de cliente, OnChartEvent

© 2000-2025, MetaQuotes Ltd.

ZeroMemory

La función reinicializa la variable que le ha sido pasada por referencia.

```
void  ZeroMemory(
   void & variable      // variable reinicializada
   );
```

Parámetros

variable

[in] [out] Variable pasada por referencia a la que hay que reinicializar (inicializar con valores cero).

Valor devuelto

No hay valor devuelto.

Nota

Si el parámetro de la función es una cadena, esta llamada va a ser equivalente a la indicación de NULL como su valor.
Para los tipos simples y sus arrays, así como para las estructuras/clases compuestos de estos tipos, esto es una simple puesta a cero.
Para los objetos que contienen cadenas y arrays dinámicos, ZeroMemory() se llama para cada uno de los elementos.
Para cualquier array que no está protegido por el modificador const, se realiza puesta a cero de todos los elementos.
Para los arrays de objetos complejos, ZeroMemory() se llama para cada uno de los elementos.

La función ZeroMemory() no se aplica a las clases con miembros protegidos o herencia.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos e incializamos la línea
   string str="Test ZeroMemory func";
//--- mostramos en el diario la línea hasta la aplicación a la misma de ZeroMemory()
   PrintFormat("The line before applying ZeroMemory() to it: '%s'",str);
//--- ponemos a cero la línea y registramos el resultado
   ZeroMemory(str);
   Print("The same line after applying ZeroMemory() to it: '",str,"'");
  /*
  Resultado:
   The line before applying ZeroMemory() to it: 'Test ZeroMemory func'
   The same line after applying ZeroMemory() to it: ''
  */
```

© 2000-2025, MetaQuotes Ltd.

```
//--- declaramos e inicializamos la variable con el tipo int
   int var=123;
//--- mostramos en el diario la línea hasta la aplicación a la misma de ZeroMemory()
   PrintFormat("\nThe integer variable before applying ZeroMemory() to it: %d",var);
//--- ponemos a cero la variable y mostramos el resultado en el diario
   ZeroMemory(var);
   PrintFormat("The same variable after applying ZeroMemory() to it: %d",var);
  /*
  Resultado:
   The integer variable before applying ZeroMemory() to it: 123
   The same variable after applying ZeroMemory() to it: 0
  */

//--- declaramos e inicializamos el array con el tipo int
   int arr[]={0,1,2,3,4,5,6,7,8,9};
//--- mostramos en el diario el array hasta la aplicación al mismo de ZeroMemory()
   Print("\nThe integer array before applying ZeroMemory() to it:");
   ArrayPrint(arr);
//--- ponemos a cero el array y mostramos el resultado en el diario
   ZeroMemory(arr);
   Print("The same array after applying ZeroMemory() to it:");
   ArrayPrint(arr);
  /*
  Resultado:
   The integer array before applying ZeroMemory() to it:
   0 1 2 3 4 5 6 7 8 9
   The same array after applying ZeroMemory() to it:
   0 0 0 0 0 0 0 0 0 0
  */

//--- declaramos una estructura de dos campos: de tipo string y entero
   struct STest
     {
      string   var_string;
      long     var_long;
     };
//--- declaramos e inicializamos un array con el tipo de estructura STest
   STest arr_struct[]={ {"0",0}, {"1",1}, {"2",2}, {"3",3} };
//--- mostramos en el diario el array de estructuras hasta la aplicación al mismo de Z
   Print("\nThe array struct before applying ZeroMemory() to it:");
   ArrayPrint(arr_struct);
//--- ponemos a cero el array de estructuras y mostramos el resultado en el diario
   ZeroMemory(arr_struct);
   Print("The same array struct after applying ZeroMemory() to it:");
   ArrayPrint(arr_struct);
  /*
  Resultado:
   The array struct before applying ZeroMemory() to it:
```

© 2000-2025, MetaQuotes Ltd.

```
      [var_string] [var_long]
 [0] "0"                   0
 [1] "1"                   1
 [2] "2"                   2
 [3] "3"                   3
 The same array struct after applying ZeroMemory() to it:
      [var_string] [var_long]
 [0] null                  0
 [1] null                  0
 [2] null                  0
 [3] null                  0
*/
}
```

© 2000-2025, MetaQuotes Ltd.

Grupo de funciones para trabajar con arrays

Los arrays como máximo pueden ser de cuatro dimensiones. La indexación de cada dimensión se realiza de 0 a *tamaño_de_dimensión-1*. En un caso particular de un array unidimensional de 50 elementos, la referencia hacia el primer elemento se verá como el array[0], hacia el último elemento - array[49].

Función	Acción
ArrayBsearch	Devuelve el índice del primer elemento encontrado en la primera dimensión del array
ArrayCopy	Copia un array al otro
ArrayCompare	Devuelve el resultado de comparación de dos arrays de tipos simples o estructuras personalizadas que no tienen objetos complejos
ArrayFree	Deja libre el búfer de cualquier array dinámico y pone el tamaño de la dimensión cero a 0.
ArrayGetAsSeries	Comprueba la dirección de indexación de un array
ArrayInitialize	Pone todos los elementos de un array numérico en el mismo valor
ArrayFill	Llena un array numérico con valor especificado
ArrayIsSeries	Comprueba si un array es una serie temporal
ArrayIsDynamic	Comprueba si un array es dinámico
ArrayMaximum	Busca un elemento con valor máximo
ArrayMinimum	Busca un elemento con valor mínimo
ArrayPrint	Muestra, en el registro, una matriz de tipo o estructura simple
ArrayRange	Devuelve el número de elementos en la dimensión especificada del array
ArrayResize	Fija nuevo tamaño en la primera dimensión del array
ArrayInsert	Inserta en la matriz-receptor el número indicado de elementos, comenzando por el índice establecido
ArrayRemove	Elimina de la matriz el número indicado de elementos, comenzando por el índice indicado
ArrayReverse	Invierte en la matriz el número indicado de elementos, comenzando por el índice indicado
ArraySetAsSeries	Establece la dirección de indexación en el array
ArraySize	Devuelve el número de elementos en el array
ArraySort	Clasificación de arrays numéricos por la primera dimensión
ArraySwap	Intercambia el contenido de dos matrices dinámicas del mismo tipo
ArrayToFP16	Permite copiar un array de tipo float o double en un array de tipo ushort con el formato establecido

© 2000-2025, MetaQuotes Ltd.

Función	Acción
ArrayToFP8	Permite copiar un array de tipo float o double en un array de tipo uchar con el formato establecido
ArrayFromFP16	Permite copiar un array de tipo ushort en un array de tipo float o double con el formato establecido
ArrayFromFP8	Permite copiar un array de tipo uchar en un array de tipo float o double con el formato establecido

© 2000-2025, MetaQuotes Ltd.

ArrayBsearch

Busca el valor especificado en el array numérico multidimensional ordenado en orden ascendente. La búsqueda se realiza por los elementos de la primera dimensión.

Para buscar en un array del tipo double

```
int  ArrayBsearch(
   const double&    array[],   // array para la búsqueda
   double           value      // lo que buscamos
   );
```

Para buscar en un array del tipo float

```
int  ArrayBsearch(
   const float&    array[],   // array para la búsqueda
   float           value      // lo que buscamos
   );
```

Para buscar en un array del tipo long

```
int  ArrayBsearch(
   const long&    array[],   // array para la búsqueda
   long           value      // lo que buscamos
   );
```

Para buscar en un array del tipo int

```
int  ArrayBsearch(
   const int&    array[],   // array para la búsqueda
   int           value      // lo que buscamos
   );
```

Para buscar en un array del tipo short

```
int  ArrayBsearch(
   const short&    array[],   // array para la búsqueda
   short           value      // lo que buscamos
   );
```

Para buscar en un array del tipo char

```
int  ArrayBsearch(
   const char&    array[],   // array para la búsqueda
   char           value      // lo que buscamos
   );
```

Parámetros

array[]

[in] Array numérico para la búsqueda.

value

© 2000-2025, MetaQuotes Ltd.

[in] Valor para la búsqueda.

Valor devuelto

Devuelve el índice del elemento encontrado. Si el valor buscado no ha sido encontrado, la función devuelve el índice del elemento más cercano por el valor.

Nota

La búsqueda binaria procesa sólo los arrays clasificados. Para clasificar un array numérico se utiliza la función ArraySort().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#property description "Script a base de los datos del indicador RSI muestra en la vent
#property description "la información sobre la frecuencia con la que el mercado se enc
#property description "y sobreventa durante el período de tiempo especificado.
//--- mostramos la ventana de los parámetros de entrada durante el arranque del script
#property script_show_inputs
//--- parámetros de entrada
input int                InpMAPeriod=14;                  // Período de Media móvil
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;     // Tipo del precio
input double             InpOversoldValue=30.0;           // Nivel de sobreventa
input double             InpOverboughtValue=70.0;         // Nivel de sobrecompra
input datetime           InpDateStart=D'2012.01.01 00:00'; // Fecha de inicio del aná
input datetime           InpDateFinish=D'2013.01.01 00:00'; // Fecha de finalización d
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   double rsi_buff[]; // array de valores del indicador
   int    size=0;     // tamaño del array
//--- obtenemos el manejador del indicador RSI
   ResetLastError();
   int rsi_handle=iRSI(Symbol(),Period(),InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- fallo al obtener el manejador del indicador
      PrintFormat("Fallo al obtener el manejador. Código del error = %d",GetLastError
      return;
     }
//--- seguimos en el ciclo hasta que el indicador no termine de calcular todos sus val
   while(BarsCalculated(rsi_handle)==-1)
     {
      //--- salimos si el usuario finaliza el trabajo del script de forma forzada
      if(IsStopped())
         return;
      //--- retardo para que al indicador le de tiempo a calcular sus valores
      Sleep(10);
     }
//--- copiamos los valores del indicador durante un período determinado
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,InpDateFinish,rsi_buff)==-1)
     {
      PrintFormat("Fallo al copiar los valores del indicador. Código del error = %d",G
      return;
     }
//--- obtenemos el tamaño del array
   size=ArraySize(rsi_buff);
//--- clasificamos el array
   ArraySort(rsi_buff);
//--- averiguamos el porcentaje de tiempo durante el que el mercado se encuentra en la
   double ovs=(double)ArrayBsearch(rsi_buff,InpOversoldValue)*100/(double)size;
//--- averiguamos el porcentaje de tiempo durante el que el mercado se encuentra en la
   double ovb=(double)(size-ArrayBsearch(rsi_buff,InpOverboughtValue))*100/(double)siz
//--- formamos cadenas para visualizar los datos
   string str="De "+TimeToString(InpDateStart,TIME_DATE)+" a "
              +TimeToString(InpDateFinish,TIME_DATE)+" el mercado estaba:";
   string str_ovb="en la zona de sobrecompra "+DoubleToString(ovb,2)+"% de tiempo";
   string str_ovs="en la zona de sobreventa "+DoubleToString(ovs,2)+"% de tiempo";
//--- mostramos los datos en el gráfico
   CreateLabel("top",5,60,str,clrDodgerBlue);
   CreateLabel("overbought",5,35,str_ovb,clrDodgerBlue);
   CreateLabel("oversold",5,10,str_ovs,clrDodgerBlue);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- redibujamos el gráfico
   ChartRedraw(0);
//--- retardo
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Mostrar comentario en la esquina inferior izquierda del gráfico  |
//+------------------------------------------------------------------+
void CreateLabel(const string name,const int x,const int y,
                 const string str,const color clr)
  {
//--- crear etiqueta
   ObjectCreate(0,name,OBJ_LABEL,0,0,0);
//--- anclar etiqueta a la esquina inferior izquierda
   ObjectSetInteger(0,name,OBJPROP_CORNER,CORNER_LEFT_LOWER);
//--- cambiamos la posición del punto de anclaje
   ObjectSetInteger(0,name,OBJPROP_ANCHOR,ANCHOR_LEFT_LOWER);
//--- distancia del punto de anclaje por el eje X
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
//--- distancia del punto de anclaje por el eje Y
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
//--- texto de la etiqueta
   ObjectSetString(0,name,OBJPROP_TEXT,str);
//--- color del texto
   ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
//--- tamaño del texto
   ObjectSetInteger(0,name,OBJPROP_FONTSIZE,12);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayCopy

Esta función copia un array al otro.

```
int  ArrayCopy(
   void&        dst_array[],          // array de destino
   const void&  src_array[],          // array de origen
   int          dst_start=0,          // índice inicial para copiar desde el array de d
   int          src_start=0,          // primer índice del array de destino
   int          count=WHOLE_ARRAY     // número de elementos
   );
```

Parámetros

dst_array[]

[out] Array de destino.

src_array[]

[in] Array de origen.

dst_start=0

[in] Índice inicial para el array de destino. Por defecto, índice inicial - 0.

src_start=0

[in] Índice inicial para el array de origen. Por defecto, índice inicial - 0.

count=WHOLE_ARRAY

[in] Número de elementos que hay que copiar. Por defecto, se copia el array entero (count=WHOLE_ARRAY).

Valor devuelto

Devuelve el número de elementos copiados.

Nota

Si count<0 o count>src_size-src_start, entonces se copia toda la parte restante del array. Los arrays se copian de izquierda a derecha. Para los arrays de serie, la posición de inicio se redefine correctamente tomando en cuenta el copiado de izquierda a derecha.

Si los arrays son de tipos diferentes, durante el copiado se intenta la transformación de cada elemento del array de origen en el tipo del array de destino. Los arrays literales se puede copiar sólo a los arrays literales. Los arrays de clases y estructuras que contienen los objetos que requieren la inicialización no se copian. Un array de estructuras puede ser copiado sólo a un array del mismo tipo.

Para los arrays dinámicos con la indexación como en las series temporales se realiza el aumento der array receptor hasta la cantidad de los datos copiados (si la cantidad de los datos copiados supera su tamaño). La reducción automática del tamaño del array receptor no se hace.

Ejemplo:

```
#property description "El indicador colorea las velas que son"
#property description "máximos y mínimos locales. La distancia del intervalo para busc
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "los valores extremos se puede establecer a través del parámetro
//--- ajustes del indicador
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot
#property indicator_label1  "Extremums"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrLightSteelBlue,clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- constante predefinida
#define INDICATOR_EMPTY_VALUE 0.0
//--- parámetros de entrada
input int InpNum=4; // Longitud de la mitad del intervalo
//--- búfers indicadores
double ExtOpen[];
double ExtHigh[];
double ExtLow[];
double ExtClose[];
double ExtColor[];
//--- variables globales
int    ExtStart=0; // índice de la primera vela que no es un extremo
int    ExtCount=0; // número de velas no- extremos en este intervalo
//+------------------------------------------------------------------+
//| Coloreado de velas no- extremos                                  |
//+------------------------------------------------------------------+
void FillCandles(const double &open[],const double &high[],
                 const double &low[],const double &close[])
  {
//--- coloreamos velas
   ArrayCopy(ExtOpen,open,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtHigh,high,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtLow,low,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtClose,close,ExtStart,ExtStart,ExtCount);
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ExtOpen);
   SetIndexBuffer(1,ExtHigh);
   SetIndexBuffer(2,ExtLow);
   SetIndexBuffer(3,ExtClose);
   SetIndexBuffer(4,ExtColor,INDICATOR_COLOR_INDEX);
//--- especificamos el valor que no va a mostrarse
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- especificamos los nombres de los búfers indicadores para mostrar en la ventana d
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- establecemos la indexación directa en las series temporales
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- variable de inicio para el cálculo de las barras
   int start=prev_calculated;
//--- para las primeras barras InpNum*2 no realizamos el cálculo
   if(start==0)
     {
      start+=InpNum*2;
      ExtStart=0;
      ExtCount=0;
     }
//--- si la barra acaba de formarse, comprobamos el siguiente extremo potencial
   if(rates_total-start==1)
      start--;
//--- índice de la barra cuyo extremo vamos a comprobar
   int ext;
//--- ciclo de cálculo de los valores del indicador
   for(int i=start;i<rates_total-1;i++)
     {
      //--- inicialmente en la barra i sin dibujar
      ExtOpen[i]=0;
      ExtHigh[i]=0;
      ExtLow[i]=0;
      ExtClose[i]=0;
      //--- índice del extremo a chequear
      ext=i-InpNum;
      //--- comprobación del máximo local
```

© 2000-2025, MetaQuotes Ltd.

```
      if(IsMax(high,ext))
        {
         //--- coloreamos la vela extremo
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=1;
         //--- las demás velas hasta el extremo marcamos con color neutral
         FillCandles(open,high,low,close);
         //--- cambiamos los valores de variables
         ExtStart=ext+1;
         ExtCount=0;
         //--- pasamos a la siguiente iteración
         continue;
        }
      //--- comprobación del mínimo local
      if(IsMin(low,ext))
        {
         //--- coloreamos la vela extremo
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=2;
         //--- las demás velas hasta el extremo marcamos con color neutral
         FillCandles(open,high,low,close);
         //--- cambiamos los valores de variables
         ExtStart=ext+1;
         ExtCount=0;
         //--- pasamos a la siguiente iteración
         continue;
        }
      //--- incrementamos el número de no-extremos en este intervalo
      ExtCount++;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Comprobamos si el elemento actual del array es un máximo local   |
//+------------------------------------------------------------------+
bool IsMax(const double &price[],const int ind)
  {
//--- variable del inicio del intervalo
   int i=ind-InpNum;
//--- variable del fin del intervalo
   int finish=ind+InpNum+1;
//--- chequeo para la primera mitad del intervalo
```

© 2000-2025, MetaQuotes Ltd.

```
   for(;i<ind;i++)
     {
      if(price[ind]<=price[i])
         return(false);
     }
//--- chequeo para la segunda mitad del intervalo
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]<=price[i])
         return(false);
     }
//--- es un extremo
   return(true);
  }
//+------------------------------------------------------------------+
//| Comprobamos si el elemento actual del array es un mínimo local   |
//+------------------------------------------------------------------+
bool IsMin(const double &price[],const int ind)
  {
//--- variable del inicio del intervalo
   int i=ind-InpNum;
//--- variable del fin del intervalo
   int finish=ind+InpNum+1;
//--- chequeo para la primera mitad del intervalo
   for(;i<ind;i++)
     {
      if(price[ind]>=price[i])
         return(false);
     }
//--- chequeo para la segunda mitad del intervalo
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]>=price[i])
         return(false);
     }
//--- es un extremo
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayCompare

Esta función devuelve el resultado de comparación de dos arrays del mismo tipo. Puede ser utilizada para comparar los arrays de tipos simples o estructuras personalizadas que no tienen objetos complejos, es decir, que no contienen cadenas de caracteres, arrays dinámicos, clases u otras estructuras que incluyen objetos complejos.

```
int  ArrayCompare(
   const void&  array1[],            // el primer array
   const void&  array2[],            // el segundo array
   int          start1=0,            // offset inicial en el primer array
   int          start2=0,            // offset inicial en el segundo array
   int          count=WHOLE_ARRAY    // número de elementos a comparar
   );
```

Parámetros

array1[]

[in] Primer array.

array2[]

[in] Segundo array.

start1=0

[in] Índice inicial del elemento en el primer array a partir del cual se empieza la comparación. Por defecto, el índice inicial es igual a 0.

start2=0

[in] Índice inicial del elemento en el segundo array a partir del cual se empieza la comparación. Por defecto, el índice inicial es igual a 0.

count=WHOLE_ARRAY

[in] Número de elementos a comparar. Por defecto, en la comparación participan todos los elementos de ambos arrays (count=WHOLE_ARRAY).

Valor devuelto

- -1, si array1[] es menor que array2[]
- 0, si array1[] y array2[] son iguales
- 1, si array1[] es mayor a array2[]
- -2, si surge un error debido a la incompatibilidad de tipos de los arrays comparados, o si los valores start1, start2 o count salen de los límites del array.

Nota

La función no va a devolver 0 (los arrays no van a considerarse iguales), si los arrays que se comparan se diferencian en su tamaño y si se indica el valor count=WHOLE_ARRAY para el caso cuando uno de los arrays es el subconjunto exacto del otro. En este caso será devuelto el resultado de comparación de estos arrays: -1, si el tamaño del array1[] es menor que el tamaño del array2[] o 1 en caso contrario.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//--- variables globales
double   ExtArrayFirst[];
double   ExtArraySecond[];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- establecemos el tamaño de los arrays
   if(ArrayResize(ExtArrayFirst,10)!=10)
     {
      Print("ArrayResize() failed for ExtArrayFirst. Error code: ",GetLastError());
      return;
     }
   if(ArrayResize(ExtArraySecond,10)!=10)
     {
      Print("ArrayResize() failed for ExtArraySecond. Error code: ",GetLastError());
      return;
     }

//--- en un ciclo, rellenamos los arrays con los valores de los índices i y j
   int total=ArraySize(ExtArrayFirst);
   for(int i=0, j=total-1; i<total; i++,j--)
     {
      //--- el array ExtArrayFirst lo rellenamos de izquierda a derecha
      //--- el array ExtArraySecond lo rellenamos de derecha a izquierda
      ExtArrayFirst[i]=i;
      ExtArraySecond[i]=j;
     }
//--- comparamos los arrays e imprimimos el resultado en el diario de registro
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
  Resultado:
   ExtArrayFirst:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   ExtArraySecond:
   9.00000 8.00000 7.00000 6.00000 5.00000 4.00000 3.00000 2.00000 1.00000 0.00000
   Result ArrayCompare(): ExtArrayFirst is smaller than ExtArraySecond (result = -1)
   */

//--- ahora invertimos los arrays
//--- en un ciclo, rellenamos los arrays con los valores de los índices i y j
   for(int i=0, j=total-1; i<total; i++,j--)
     {
      //--- el array ExtArrayFirst lo rellenamos de derecha a izquierda
      //--- el array ExtArraySecond lo rellenamos de izquierda a derecha
      ExtArrayFirst[i]=j;
      ExtArraySecond[i]=i;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- comparamos los arrays e imprimimos el resultado en el diario de registro
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
  Resultado:
   ExtArrayFirst:
   9.00000 8.00000 7.00000 6.00000 5.00000 4.00000 3.00000 2.00000 1.00000 0.00000
   ExtArraySecond:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   Result ArrayCompare(): ExtArrayFirst is larger than ExtArraySecond (result = 1)
   */

//--- ahora rellanamos los arrays en una dirección
//--- en un ciclo, rellenamos los arrays con los valores del índice i
   for(int i=0; i<total; i++)
     {
      //--- ambos arrays los rellenamos de izquierda a derecha
      ExtArrayFirst[i]=i;
      ExtArraySecond[i]=i;
     }
//--- comparamos los arrays e imprimimos el resultado en el diario de registro
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
  Resultado:
   ExtArrayFirst:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   ExtArraySecond:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   Result ArrayCompare(): ExtArrayFirst and ExtArraySecond are equal (result = 0)
   */
  }
//+------------------------------------------------------------------+
//| Comparar y mostrar resultado                                     |
//+------------------------------------------------------------------+
void ArrayComparePrint(const double &array1[], const double &array2[])
  {
   //--- imprimimos el encabezado y el contenido de los arrays
   Print("ExtArrayFirst:");
   ArrayPrint(array1);
   Print("ExtArraySecond:");
   ArrayPrint(array2);
   //--- comparamos los arrays e imprimimos el resultado de la comparación
   int    res=ArrayCompare(array1,array2);
   string res_str=(res>0 ? "ExtArrayFirst is larger than ExtArraySecond" : res<0 ? "Ex
   PrintFormat("Result ArrayCompare(): %s (result = %d)\n",res_str,res);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

ArrayFree

Esta función deja libre el búfer de cualquier array dinámico y pone el tamaño de la dimensión cero a 0.

```
void  ArrayFree(
   void&  array[]      // array
   );
```

Parámetros

array[]

[in] Array dinámico.

Valor devuelto

No hay valor devuelto.

Nota

La necesidad de utilizar la función ArrayFree() durante la escritura de los scripts e indicadores puede surgir con poca frecuencia. Es que cuando el trabajo del script se finaliza la memoria utilizada se libera en seguida, el trabajo principal con los arrays en los indicadores personalizados consiste en el acceso a los búferes de indicadores cuyos tamaños se controlan automáticamente por el subsistema ejecutivo del terminal.

Si en el programa es necesario administrar la memoria manualmente en unas condiciones dinámicas complicadas, la función ArrayFree()permite al usuario liberar la memoria ocupada por un array dinámico ya innecesario de forma explícita e inmediata.

Ejemplo:

```
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
#include <Controls\Label.mqh>
#include <Controls\ComboBox.mqh>
//--- constantes predefinidas
#define X_START 0
#define Y_START 0
#define X_SIZE 280
#define Y_SIZE 300
//+------------------------------------------------------------------+
//| Clase de diálogo para trabajar con la memoria                    |
//+------------------------------------------------------------------+
class CMemoryControl : public CAppDialog
  {
private:
   //--- tamaño del array
   int               m_arr_size;
   //--- arrays
   char              m_arr_char[];
   int               m_arr_int[];
   float             m_arr_float[];
```

© 2000-2025, MetaQuotes Ltd.

```
   double            m_arr_double[];
   long              m_arr_long[];
   //--- etiquetas
   CLabel            m_lbl_memory_physical;
   CLabel            m_lbl_memory_total;
   CLabel            m_lbl_memory_available;
   CLabel            m_lbl_memory_used;
   CLabel            m_lbl_array_size;
   CLabel            m_lbl_array_type;
   CLabel            m_lbl_error;
   CLabel            m_lbl_change_type;
   CLabel            m_lbl_add_size;
   //--- botones
   CButton           m_button_add;
   CButton           m_button_free;
   //--- cajas combinadas
   CComboBox         m_combo_box_step;
   CComboBox         m_combo_box_type;
   //--- valor actual del tipo del array de la caja combinada
   int               m_combo_box_type_value;

public:
                     CMemoryControl(void);
                    ~CMemoryControl(void);
   //--- método de creación del objeto de la clase
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- manejador de eventos del gráfico
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crear etiqueta
   bool              CreateLabel(CLabel &lbl,const string name,const int x,const int y
   //--- crear elementos de control
   bool              CreateButton(CButton &button,const string name,const int x,const
   bool              CreateComboBoxStep(void);
   bool              CreateComboBoxType(void);
   //--- manejadores de eventos
   void              OnClickButtonAdd(void);
   void              OnClickButtonFree(void);
   void              OnChangeComboBoxType(void);
   //--- métodos de trabajo con array actual
   void              CurrentArrayFree(void);
   bool              CurrentArrayAdd(void);
  };
//+------------------------------------------------------------------+
//| Liberación de memoria del array actual                           |
//+------------------------------------------------------------------+
void CMemoryControl::CurrentArrayFree(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- reiniciar el tamaño del array
   m_arr_size=0;
//--- liberación del array
   if(m_combo_box_type_value==0)
      ArrayFree(m_arr_char);
   if(m_combo_box_type_value==1)
      ArrayFree(m_arr_int);
   if(m_combo_box_type_value==2)
      ArrayFree(m_arr_float);
   if(m_combo_box_type_value==3)
      ArrayFree(m_arr_double);
   if(m_combo_box_type_value==4)
      ArrayFree(m_arr_long);
  }
//+------------------------------------------------------------------+
//| Intento de agregar memoria para array actual                     |
//+------------------------------------------------------------------+
bool CMemoryControl::CurrentArrayAdd(void)
  {
//--- salir si el tamaño de la memoria utilizada supera el tamaño de la memoria física
   if(TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL)/TerminalInfoInteger(TERMINAL_MEMOR
      return(false);
//--- intento de adjudicación de la memoria de acuerdo con el tipo actual
   if(m_combo_box_type_value==0 && ArrayResize(m_arr_char,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==1 && ArrayResize(m_arr_int,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==2 && ArrayResize(m_arr_float,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==3 && ArrayResize(m_arr_double,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==4 && ArrayResize(m_arr_long,m_arr_size)==-1)
      return(false);
//--- memoria adjudicada
   return(true);
  }
//+------------------------------------------------------------------+
//| Manejo de eventos                                                |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CMemoryControl)
ON_EVENT(ON_CLICK,m_button_add,OnClickButtonAdd)
ON_EVENT(ON_CLICK,m_button_free,OnClickButtonFree)
ON_EVENT(ON_CHANGE,m_combo_box_type,OnChangeComboBoxType)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CMemoryControl::CMemoryControl(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CMemoryControl::~CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------+
//| Método de creación del objeto de la clase                        |
//+------------------------------------------------------------------+
bool CMemoryControl::Create(const long chart,const string name,const int subwin,
                            const int x1,const int y1,const int x2,const int y2)
  {
//--- creación del objeto de la clase basa
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- preparación de las cadenas para las etiquetas
   string str_physical="Memory physical = "+(string)TerminalInfoInteger(TERMINAL_MEMOR
   string str_total="Memory total = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_TOTA
   string str_available="Memory available = "+(string)TerminalInfoInteger(TERMINAL_MEM
   string str_used="Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_USED)+
//--- crear etiquetas
   if(!CreateLabel(m_lbl_memory_physical,"physical_label",X_START+10,Y_START+5,str_phy
      return(false);
   if(!CreateLabel(m_lbl_memory_total,"total_label",X_START+10,Y_START+30,str_total,12
      return(false);
   if(!CreateLabel(m_lbl_memory_available,"available_label",X_START+10,Y_START+55,str_
      return(false);
   if(!CreateLabel(m_lbl_memory_used,"used_label",X_START+10,Y_START+80,str_used,12,cl
      return(false);
   if(!CreateLabel(m_lbl_array_type,"type_label",X_START+10,Y_START+105,"Array type =
      return(false);
   if(!CreateLabel(m_lbl_array_size,"size_label",X_START+10,Y_START+130,"Array size =
      return(false);
   if(!CreateLabel(m_lbl_error,"error_label",X_START+10,Y_START+155,"",12,clrRed))
      return(false);
   if(!CreateLabel(m_lbl_change_type,"change_type_label",X_START+10,Y_START+185,"Chang
      return(false);
   if(!CreateLabel(m_lbl_add_size,"add_size_label",X_START+10,Y_START+210,"Add to arra
      return(false);
//--- crear elementos de control
   if(!CreateButton(m_button_add,"add_button",X_START+15,Y_START+245,"Add",12,clrBlue)
      return(false);
   if(!CreateButton(m_button_free,"free_button",X_START+75,Y_START+245,"Free",12,clrBl
      return(false);
   if(!CreateComboBoxType())
      return(false);
   if(!CreateComboBoxStep())
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- inicialización de variable
   m_arr_size=0;
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Crear el botón                                                   |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateButton(CButton &button,const string name,const int x,
                                  const int y,const string str,const int font_size,
                                  const int clr)
  {
//--- crear el botón
   if(!button.Create(m_chart_id,name,m_subwin,x,y,x+50,y+20))
      return(false);
//--- texto
   if(!button.Text(str))
      return(false);
//--- tamaño de la fuente
   if(!button.FontSize(font_size))
      return(false);
//--- color de la etiqueta
   if(!button.Color(clr))
      return(false);
//--- agregamos el botón a los elementos de control
   if(!Add(button))
      return(false);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Crear la caja combinada para el tamaño del array                 |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxStep(void)
  {
//--- crear la caja combinada
   if(!m_combo_box_step.Create(m_chart_id,"step_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- agregar elementos a la caja combinada
   if(!m_combo_box_step.ItemAdd("100 000",100000))
      return(false);
   if(!m_combo_box_step.ItemAdd("1 000 000",1000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("10 000 000",10000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("100 000 000",100000000))
      return(false);
//--- establecer el elemento actual de la caja combinada
   if(!m_combo_box_step.SelectByValue(1000000))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- agregamos la caja combinada a los elementos de control
   if(!Add(m_combo_box_step))
      return(false);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Crear la caja combinada para el tipo del array                   |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxType(void)
  {
//--- crear la caja combinada
   if(!m_combo_box_type.Create(m_chart_id,"type_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- agregar elementos a la caja combinada
   if(!m_combo_box_type.ItemAdd("char",0))
      return(false);
   if(!m_combo_box_type.ItemAdd("int",1))
      return(false);
   if(!m_combo_box_type.ItemAdd("float",2))
      return(false);
   if(!m_combo_box_type.ItemAdd("double",3))
      return(false);
   if(!m_combo_box_type.ItemAdd("long",4))
      return(false);
//--- establecer el elemento actual de la caja combinada
   if(!m_combo_box_type.SelectByValue(3))
      return(false);
//--- memorizamos el elemento actual de la caja combinada
   m_combo_box_type_value=3;
//--- agregamos la caja combinada a los elementos de control
   if(!Add(m_combo_box_type))
      return(false);
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Crear etiqueta                                                   |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateLabel(CLabel &lbl,const string name,const int x,
                                 const int y,const string str,const int font_size,
                                 const int clr)
  {
//--- crear etiqueta
   if(!lbl.Create(m_chart_id,name,m_subwin,x,y,0,0))
      return(false);
//--- texto
   if(!lbl.Text(str))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- tamaño de la fuente
   if(!lbl.FontSize(font_size))
      return(false);
//--- color
   if(!lbl.Color(clr))
      return(false);
//--- agregamos etiqueta a los elementos de control
   if(!Add(lbl))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Manejador del evento de pulsación del botón "Add"                |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonAdd(void)
  {
//--- aumentamos el tamaño del array
   m_arr_size+=(int)m_combo_box_step.Value();
//--- intentamos adjudicar la memoria para el array actual
   if(CurrentArrayAdd())
     {
      //--- memoria adjudicada, mostramos el estado actual sobre la pantalla
      m_lbl_memory_available.Text("Memory available = "+(string)TerminalInfoInteger(TE
      m_lbl_memory_used.Text("Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEM
      m_lbl_array_size.Text("Array size = "+IntegerToString(m_arr_size));
      m_lbl_error.Text("");
     }
   else
     {
      //--- no se ha podido adjudicar la memoria, mostramos mensaje sobre el error
      m_lbl_error.Text("Array is too large, error!");
      //--- devolvemos el tamaño anterior del array
      m_arr_size-=(int)m_combo_box_step.Value();
     }
  }
//+------------------------------------------------------------------+
//| Manejador del evento de pulsación del botón "Free"               |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonFree(void)
  {
//--- liberamos la memoria del array actual
   CurrentArrayFree();
//--- mostramos el estado actual sobre la pantalla
   m_lbl_memory_available.Text("Memory available = "+(string)TerminalInfoInteger(TERMI
   m_lbl_memory_used.Text("Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEMORY
   m_lbl_array_size.Text("Array size = 0");
   m_lbl_error.Text("");
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Manejador del evento de la modificación de caja combinada        |
//+------------------------------------------------------------------+
void CMemoryControl::OnChangeComboBoxType(void)
  {
//--- prueba, si se ha cambiado el tipo del array
   if(m_combo_box_type.Value()!=m_combo_box_type_value)
     {
      //--- liberamos la memoria del array actual
      OnClickButtonFree();
      //--- trabajamos con otro tipo del array
      m_combo_box_type_value=(int)m_combo_box_type.Value();
      //--- mostramos nuevo tipo del array en la pantalla
      if(m_combo_box_type_value==0)
         m_lbl_array_type.Text("Array type = char");
      if(m_combo_box_type_value==1)
         m_lbl_array_type.Text("Array type = int");
      if(m_combo_box_type_value==2)
         m_lbl_array_type.Text("Array type = float");
      if(m_combo_box_type_value==3)
         m_lbl_array_type.Text("Array type = double");
      if(m_combo_box_type_value==4)
         m_lbl_array_type.Text("Array type = long");
     }
  }
//--- objeto de la clase CMemoryControl
CMemoryControl ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crear diálogo
   if(!ExtDialog.Create(0,"MemoryControl",0,X_START,Y_START,X_SIZE,Y_SIZE))
      return(INIT_FAILED);
//--- iniciar
   ExtDialog.Run();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ExtDialog.Destroy(reason);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayGetAsSeries

Esta función comprueba la dirección de indexación de un array.

```
bool  ArrayGetAsSeries(
   const void&  array[]    // array a comprobar
   );
```

Parámetros

array

[in] Array comprobado.

Valor devuelto

Devuelve true, si el array especificado tiene puesta la bandera AS_SERIES, es decir, el acceso al array se realiza al revés, como en una serie temporal. Una serie temporal se diferencia de un array usual en que la indexación de los elementos de la serie temporal se realiza del fin del array al inicio (de los datos más recientes a los más antiguos).

Nota

Para comprobar si un array pertenece a una serie temporal hay que usar la función ArrayIsSeries(). Los arrays de los datos de precio que han sido pasados como los parámetros de entrada a la función OnCalculate(), no han de tener obligatoriamente la dirección de indexación como las series temporales. La función ArraySetAsSeries() puede establecer la dirección de indexación necesaria.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#property description "El indicador calcula los valores absolutos de diferencia entre
#property description "Open y Close o High y Low, y los visualiza en una subventana ir
#property description "en forma de un histograma."
//--- ajustes del indicador
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- parámetros de entrada
input bool InpAsSeries=true; // Dirección de indexación en el búfer de indicadores
input bool InpPrices=true;   // Precios para el cálculo (true - Open,Close; false - Hi
//--- búfer de indicadores
double ExtBuffer[];
//+------------------------------------------------------------------+
//| Cálculo de valores del indicador                                 |
//+------------------------------------------------------------------+
void CandleSizeOnBuffer(const int rates_total,const int prev_calculated,
                        const double &first[],const double &second[],double &buffer[])
  {
//--- variable inicial para el cálculo de barras
   int start=prev_calculated;
//--- si los valores del indicador ya han sido calculados en el tick anterior, entonce
   if(prev_calculated>0)
      start--;
//--- definimos la dirección de la indexación en los arrays
   bool as_series_first=ArrayGetAsSeries(first);
   bool as_series_second=ArrayGetAsSeries(second);
   bool as_series_buffer=ArrayGetAsSeries(buffer);
//--- si hace falta, sustituimos la dirección de la indexación con la recta
   if(as_series_first)
      ArraySetAsSeries(first,false);
   if(as_series_second)
      ArraySetAsSeries(second,false);
   if(as_series_buffer)
      ArraySetAsSeries(buffer,false);
//--- calculamos los valores del indicador
   for(int i=start;i<rates_total;i++)
      buffer[i]=MathAbs(first[i]-second[i]);
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- enlace de los búferes de indicadores
   SetIndexBuffer(0,ExtBuffer);
//--- fijamos la dirección de la indexación en el búfer de indicadores
   ArraySetAsSeries(ExtBuffer,InpAsSeries);
//--- comprobamos para qué tipo de precios se calcula el indicador
   if(InpPrices)
     {
      //--- precios Open y Close
      PlotIndexSetString(0,PLOT_LABEL,"BodySize");
      //--- determinamos el color del indicador
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrOrange);
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- precios High y Low
      PlotIndexSetString(0,PLOT_LABEL,"ShadowSize");
      //--- determinamos el color del indicador
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrDodgerBlue);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- cálculo del indicador en función del valor de la bandera
   if(InpPrices)
      CandleSizeOnBuffer(rates_total,prev_calculated,open,close,ExtBuffer);
   else
      CandleSizeOnBuffer(rates_total,prev_calculated,high,low,ExtBuffer);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Acceso a las series temporales, ArraySetAsSeries

© 2000-2025, MetaQuotes Ltd.

ArrayInitialize

Esta función inicializa un array numérico usando un valor especificado.

For initialization of an array of char type

```
int  ArrayInitialize(
   char    array[],     // array inicializado
   char    value        // valor que va a ser puesto
   );
```

For initialization of an array of short type

```
int  ArrayInitialize(
   short   array[],     // array inicializado
   short   value        // valor que va a ser puesto
   );
```

For initialization of an array of int type

```
int  ArrayInitialize(
   int     array[],     // array inicializado
   int     value        // valor que va a ser puesto
   );
```

For initialization of an array of long type

```
int  ArrayInitialize(
   long    array[],     // array inicializado
   long    value        // valor que va a ser puesto
   );
```

For initialization of an array of float type

```
int  ArrayInitialize(
   float   array[],     // array inicializado
   float   value        // valor que va a ser puesto
   );
```

For initialization of an array of double type

```
int  ArrayInitialize(
   double  array[],     // array inicializado
   double  value        // valor que va a ser puesto
   );
```

For initialization of an array of bool type

```
int  ArrayInitialize(
   bool    array[],     // array inicializado
   bool    value        // valor que va a ser puesto
   );
```

© 2000-2025, MetaQuotes Ltd.

For initialization of an array of uint type

```
int  ArrayInitialize(
   uint    array[],     // array inicializado
   uint    value        // valor que va a ser puesto
   );
```

Parámetros

array[]

[out] Array numérico que hay que inicializar.

value

[in] Nuevo valor que hay que asignar a todos los elementos del array.

Valor devuelto

Número de elementos.

Nota

La función ArrayResize() permite establecer el tamaño de un array con una cierta reserva para su futura expansión sin redistribución física de la memoria. Esto se implementa para mejorar el rendimiento, puesto que las operaciones de redistribución de la memoria son bastante lentas.

La inicialización del array usando la expresión ArrayInitialize(array, init_val) no significa la inicialización con el mismo valor de los elementos de la reserva adjudicada para este array. Con las futuras expansiones del tamaño del array *array* mediante la función ArrayResize() dentro de los márgenes de la reserva actual, al final del array se añadirán los elementos cuyos valores no estarán definidos, y en mayoría de las ocasiones no van a ser iguales a *init_val*.

Ejemplo:

```
void OnStart()
  {
//--- array dinámico
   double array[];
//--- vamos a establecer el tamaño del array para 100 elementos y reservar el búfer pa
   ArrayResize(array,100,10);
//--- inicializamos los elementos del array con el valor EMPTY_VALUE=DBL_MAX
   ArrayInitialize(array,EMPTY_VALUE);
   Print("Valores de los últimos 10 elementos del array después de la inicialización")
   for(int i=90;i<100;i++) printf("array[%d] = %G",i,array[i]);
//--- aumentamos el tamaño del array por 5 elementos
   ArrayResize(array,105);
   Print("Valores de los últimos 10 elementos del array después de ArrayResize(array,1
//--- los valores de 5 últimos elementos han sido recibidos desde el buffer de reserva
   for(int i=95;i<105;i++) printf("array[%d] = %G",i,array[i]);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayFill

Llena un array numérico con valor especificado.

```
void  ArrayFill(
   void&  array[],       // array
   int    start,         // índice del elemento inicial
   int    count,         // número de elementos
   void   value          // valor con el que se llena el array
   );
```

Parámetros

array[]

[out] Array del tipo simple (char, uchar, short, ushort, int, uint, long, ulong, bool, color, datetime, float, double).

start

[in] Índice del elemento inicial (a partir del cual empezamos a llenar). En este caso, la bandera de series establecida se ignora.

count

[in] Número de elementos a rellenar.

value

[in] Valor con el que se llena el array.

Valor devuelto

No hay valor devuelto.

Nota

Cuando se llama a la función ArrayFill(), siempre se sobreentiende la dirección habitual de la indexación (de izquierda a derecha). Es decir, el cambio del orden de acceso a los elementos del array mediante la función ArraySetAsSeries() no se toma en consideración.

Un array multidimensional se representa como unidimensional cuando se procesa por la función ArrayFill(). Por ejemplo, el array array[2][4] se procesa como array[8]. Por eso a la hora de trabajar con este array se puede especificar el índice del elemento inicial igual a 5. De esta manera, la llamada a ArrayFill(array, 5, 2, 3.14) para el array array[2][4] va a llenar los elementos del array array[1][1] y array[1][2] con el valor 3.14.

Ejemplo:

```
void OnStart()
  {
//--- declaramos un array dinámico
   int a[];
//--- establecemos el tamaño
   ArrayResize(a,10);
//--- llenamos 5 elementos iniciales con el valor 123
   ArrayFill(a,0,5,123);
//--- llenamos 5 elementos (a partir del quinto) con el valor 456
   ArrayFill(a,5,5,456);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos los valores de todos los elementos
   for(int i=0;i<ArraySize(a);i++) printf("a[%d] = %d",i,a[i]);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayIsDynamic

La función comprueba si el array es dinámico.

```
bool  ArrayIsDynamic(
   const void&  array[]    // array comprobado
   );
```

Parámetros

array[]

[in] Array que se comprueba.

Valor devuelto

Devuelve true, si el array indicado es dinámico, de lo contrario devuelve false.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#property description "Este indicador no calcula valores, sino intenta aplicar una vez
#property description "la llamada de la función ArrayFree() a tres arrays: dinámico, e
#property description "búfer de indicadores. Los resultados se muestran en el diario d
//--- ajustes del indicador
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- variables globales
double ExtDynamic[];   // array dinámico
double ExtStatic[100]; // array estático
bool   ExtFlag=true;   // bandera
double ExtBuff[];      // búfer de indicadores
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- adjudicamos la memoria para el array
   ArrayResize(ExtDynamic,100);
//--- indicator buffers mapping
   SetIndexBuffer(0,ExtBuff);
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])
  {
//--- realizamos el análisis singular
   if(ExtFlag)
     {
      //--- tratamos de liberar la memoria para los arrays
      //--- 1. Array dinámico
      Print("+============================+");
      Print("1. Cheque del array dinámico:");
      Print("Tamaño antes de liberar memoria = ",ArraySize(ExtDynamic));
      Print("Es un array dinámico = ",ArrayIsDynamic(ExtDynamic) ? "Sí" : "No");
      //--- tratamos de liberar la memoria del array
      ArrayFree(ExtDynamic);
      Print("Tamaño después de liberar memoria = ",ArraySize(ExtDynamic));
      //--- 2. Array estático
      Print("2. Chequeo del array estático:");
      Print("Tamaño antes de liberar memoria = ",ArraySize(ExtStatic));
      Print("Es un array estático = ",ArrayIsDynamic(ExtStatic) ? "Sí" : "No");
      //--- tratamos de liberar la memoria del array
      ArrayFree(ExtStatic);
      Print("Tamaño después de liberar memoria = ",ArraySize(ExtStatic));
      //--- 3. Búfer de indicadores
      Print("3. Chequeo del búfer de indicadores:");
      Print("Tamaño antes de liberar memoria = ",ArraySize(ExtBuff));
      Print("Es un array dinámico = ",ArrayIsDynamic(ExtBuff) ? "Sí" : "No");
      //--- tratamos de liberar la memoria del array
      ArrayFree(ExtBuff);
      Print("Tamaño después de liberar memoria = ",ArraySize(ExtBuff));
      //--- cambiamos el valor de la bandera
      ExtFlag=false;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Acceso a las series temporales e indicadores

© 2000-2025, MetaQuotes Ltd.

ArrayIsSeries

Esta función comprueba si el array es una serie temporal.

```
bool  ArrayIsSeries(
   const void&  array[]    // array comprobado
   );
```

Parámetros

array[]

[in] Array que se comprueba.

Valor devuelto

Devuelve true, si el array comprobado es un array-serie temporal, de lo contrario devuelve false. Los arrays pasados como un parámetro a la función OnCalculate(), tienen que ser comprobados para el orden de acceso a los elementos del array con la función ArrayGetAsSeries().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   if(ArrayIsSeries(open))
      Print("open[] is timeseries");
   else
      Print("open[] is not timeseries!!!");
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Acceso a las series temporales e indicadores

© 2000-2025, MetaQuotes Ltd.

ArrayMaximum

Busca el elemento máximo en la primera dimensión del array numérico multidimensional.

```
int  ArrayMaximum(
   const void&   array[],            // array para la búsqueda
   int           start=0,            // a partir de qué índice empezamos a buscar
   int           count=WHOLE_ARRAY   // número de elementos a comprobar
   );
```

Parámetros

array[]

[in] Array numérico donde se realiza la búsqueda.

start=0

[in] Índice de partida para la búsqueda.

count=WHOLE_ARRAY

[in] Número de elementos para la búsqueda. Por defecto, buscamos en todo el array (count=WHOLE_ARRAY).

Valor devuelto

La función devuelve el índice del elemento encontrado teniendo en cuenta la serie del array. En caso del fallo la función devuelve -1.

Nota

El elemento mínimo se busca tomando en cuenta el valor de la bandera AS_SERIES.

Las funciones ArrayMaximum y ArrayMinimum reciben como parámetro el array de cualquiera dimensión, la búsqueda se realiza sólo para la primera (cero) dimensión.

Ejemplo:

```
#property description "El indicador muestra las velas del mayor período en el período
//--- ajustes del indicador
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- plot 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- plot 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
```

© 2000-2025, MetaQuotes Ltd.

```
//---- plot 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- plot 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- plot 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- constante predefinida
#define INDICATOR_EMPTY_VALUE 0.0
//--- parámetros de entrada
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Período para el cálculo del
input datetime        InpDateStart=D'2013.01.01 00:00'; // Fecha de inicio del análisi
//--- búferes de indicadores para las velas bajistas
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- búferes de indicadores para las velas alcistas
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- variables globales
datetime ExtTimeBuff[];      // búfer de tiempo del período mayor
int      ExtSize=0;          // tamaño del búfer de tiempo
int      ExtCount=0;         // índice dentro del búfer de tiempo
int      ExtStartPos=0;      // posición inicial para el cálculo del indicador
bool     ExtStartFlag=true;  // bandera auxiliar para recibir la posición inicial
datetime ExtCurrentTime[1];  // última hora de formación de la barra desde el período
datetime ExtLastTime;        // última hora desde el período mayor para el que se ha h
bool     ExtBearFlag=true;   // bandera para determinar el orden de escritura de datos
bool     ExtBullFlag=true;   // bandera para determinar el orden de escritura de datos
int      ExtIndexMax=0;      // índice del elemento máximo del array
int      ExtIndexMin=0;      // índice del elemento mínimo del array
int      ExtDirectionFlag=0; // dirección de movimiento del precio para la vela actual
//--- espacio entre el precio de apertura y cierre de la vela para una correcta repres
```

© 2000-2025, MetaQuotes Ltd.

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//| Coloreado de la parte básica de la vela                          |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- buscamos los índices del elemento máximo y del mínimo en los arrays
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // el máximo en High
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // el mínimo en Low
//--- determinamos el número de barras desde el período actual que vamos a colear
   int count=fill_index-start+1;
//--- si el precio de cierre de la primera barra supera el precio de cierre de la últi
   if(open[start]>close[last])
     {
      //--- si hasta este momento la vela era bajista, entonces limpiamos los valores
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- vela bajista
      ExtDirectionFlag=-1;
      //--- formamos la vela
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- salida de la función
      return;
     }
//--- si el precio de cierre de la primera barra es inferior al precio de cierre de la
   if(open[start]<close[last])
     {
      //--- si hasta este momento la vela era alcistas, entonces limpiamos los valores
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- vela alcista
      ExtDirectionFlag=1;
      //--- formamos la vela
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- salida de la función
      return;
     }
//--- si se encuentra en esta parte de la función, el precio de apertura de la primera
//--- precio de cierre de la última barra; vamos a considerar esta vela bajista
//--- si hasta este momento la vela era bajista, entonces limpiamos los valores de los
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- vela bajista
   ExtDirectionFlag=-1;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si los precios de cierre y de apertura son iguales, utilizamos el desplazamiento
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//| Coloreado de la punta de la vela                                 |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- no dibujamos en caso de sólo una barra
   if(last-start==0)
      return;
//--- si el precio de cierre de la primera barra supera el precio de cierre de la últi
   if(open[start]>close[last])
     {
      //--- formamos la punta de la vela
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- salida de la función
      return;
     }
//--- si el precio de cierre de la primera barra es inferior al precio de cierre de la
   if(open[start]<close[last])
     {
      //--- formamos la punta de la vela
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- salida de la función
      return;
     }
//--- si se encuentra en esta parte de la función, el precio de apertura de la primera
//--- precio de cierre de la última barra; vamos a considerar esta vela bajista
//--- formamos la punta de la vela
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEm
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- chequeo del período de tiempo del indicador
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- visualización de los precios en el primer plano
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- enlace de los búferes de indicadores
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- fijamos algunos valores de propiedades para construir el indicador
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // tipo de construcción grá
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // estilo de la línea
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // grosor de la línea
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- si todavía no hay barras calculadas,
   if(prev_calculated==0)
     {
      //--- obtenemos la hora de aparición de las barras desde el período mayor
      if(!GetTimeData())
         return(0);
     }
//--- establecemos la dirección directa para la indexación
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- variable de inicio para el cálculo de las barras
   int start=prev_calculated;
//--- si la barra se está formando, volvemos a calcular el valor del indicador sobre e
   if(start!=0 && start==rates_total)
      start--;
//--- ciclo de cálculo de los valores del indicador
   for(int i=start;i<rates_total;i++)
     {
      //--- llenamos los elementos "i" de los búferes de indicadores con valores vacío
      FillIndicatorBuffers(i);
      //--- hacemos el cálculo para las barras a partir de la fecha InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- definimos por primera vez la posición a partir de la cual empezamos a r
         if(ExtStartFlag)
           {
            //--- recordamos el número de la barra inicial
            ExtStartPos=i;
            //--- determinamos la primera fecha desde el período mayor que supera time
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- cambiamos el valor de la bandera para no volver a entrar en este blo
            ExtStartFlag=false;
           }
         //--- comprobamos si hay más elementos en el array
         if(ExtCount<ExtSize)
           {
            //--- esperamos hasta que el valor de tiempo del período actual alcance el
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- dibujamos la parte básica de la vela (sin colorear entre la últir
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,ExtI
               //--- coloreamos la punta de la vela (área entre la última barra y la p
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
```

© 2000-2025, MetaQuotes Ltd.

```
                //--- movemos la posición inicial para dibujar la siguiente barra
                ExtStartPos=i;
                //--- aumentamos el contador del array
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- anulamos los valores del error
             ResetLastError();
             //--- obtenemos la última fecha del período mayor
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Error del copiado de datos, código = ",GetLastError());
                return(0);
               }
             //--- si nueva fecha es mayor, terminamos la formación de la vela
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- limpiamos el área entre la última barra y la penúltima en los búf
                ClearEndOfBodyMain(i-1);
                //--- coloreamos esta área utilizando los búferes de indicadores auxili
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- movemos la posición inicial para dibujar la siguiente barra
                ExtStartPos=i;
                //--- reseteamos la bandera de dirección del precio
                ExtDirectionFlag=0;
                //--- recordamos la última fecha nueva
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- formamos la vela
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Prueba de la corrección del período del indicador introducido    |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- el período del indicador tiene que ser más grande que el período de tiempo (time
```

© 2000-2025, MetaQuotes Ltd.

```
   if(current_period>=high_period)
     {
      Print("¡Error! ¡El valor del período del indicador tiene que superar al valor de
      return(false);
     }
//--- si el período del indicador es una semana o un mes, entonces el período es corre
   if(high_period>32768)
      return(true);
//--- convertimos los valores de los períodos a los minutos
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- el período del indicador debe de ser múltiple del período de tiempo en el que se
   if(high_period%current_period!=0)
     {
      Print("¡Error! ¡El valor del período del indicador tiene que múltiple del valor
      return(false);
     }
//--- el período del indicador tiene que superar al período de tiempo (time frame) en
   if(high_period/current_period<3)
     {
      Print("¡Error! ¡El valor del período del indicador tiene que superar al valor de
      return(false);
     }
período del indicador es correcto para el período de tiempo actual
   return(true);
  }
//+------------------------------------------------------------------+
//| Recepción de datos de tiempo desde el período de tiempo mayor    |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- resetear el valor del error
   ResetLastError();
//--- copiamos todos los datos para la hora actual
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- obtenemos el código del error
      int code=GetLastError();
      //--- imprimimos el texto del error
      PrintFormat("¡Error al copiar datos! %s",code==4401
                  ? "¡El historial se sigue cargando!"
                  : "Código = "+IntegerToString(code));
      //--- devolvemos false para volver a intentar cargar datos
      return(false);
     }
//--- obtenemos el tamaño del array
   ExtSize=ArraySize(ExtTimeBuff);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ponemos el índice del ciclo para el array igual a cero
   ExtCount=0;
//--- ponemos la posición de la vela actual en este período de tiempo igual a cero
   ExtStartPos=0;
   ExtStartFlag=true;
//--- recordamos el último valor de tiempo desde el período de tiempo mayor
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función forma la parte básica de la vela. Dependiendo del valor de|
//| la bandera, la función determina qué datos y en qué arrays deben |
//| escribirse para una visualización correcta.                      |
//+------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- comprobamos el valor de la bandera
   if(flag)
     {
      //--- formamos el cuerpo de la vela
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- formamos la sombra de la vela
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- formamos el cuerpo de la vela
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- formamos la sombra de la vela
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------+
//| La función forma la punta de la vela. Dependiendo del valor de la bandera,  |
//| la función determina qué datos y en qué arrays deben             |
//| escribirse para una visualización correcta.                      |
//+------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- comprobamos el valor de la bandera
```

© 2000-2025, MetaQuotes Ltd.

```
   if(flag)
     {
      //--- formamos el fin del cuerpo de la vela
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- formamos el fin de la sombra de la vela
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- cambiamos el valor de la bandera al opuesto
      flag=false;
     }
   else
     {
      //--- formamos el fin del cuerpo de la vela
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- formamos el fin de la sombra de la vela
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- cambiamos el valor de la bandera al opuesto
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Limpiamos la punta de la vela (área entre la última barra y la penúltima     |
//| barra)                                                           |
//+------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------+
//| Limpieza de la vela                                              |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- prueba
   if(count!=0)
     {
      //--- llenamos los búferes indicadores con valores vacíos
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------+
//| Formación de la parte básica de la vela                          |
//+------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- prueba
   if(count!=0)
     {
      //--- llenamos los búferes indicadores con valores
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| Formación de la punta de la vela                                 |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- llenamos los búferes indicadores con valores
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Llenar los elementos "i" de los búferes de indicadores con valores vacíos|
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- establecemos un valor vacío en la cédula de búferes de indicadores
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayMinimum

Busca el elemento mínimo en la primera dimensión del array numérico multidimensional.

```
int  ArrayMinimum(
   const void&   array[],             // array para la búsqueda
   int           start=0,             // a partir de qué índice empezamos a buscar
   int           count=WHOLE_ARRAY    // número de elementos a comprobar
   );
```

Parámetros

array[]

[in] Array numérico donde se realiza la búsqueda.

start=0

[in] Índice de partida para la búsqueda.

count=WHOLE_ARRAY

[in] Número de elementos para la búsqueda. Por defecto, buscamos en todo el array (count=WHOLE_ARRAY).

Valor devuelto

La función devuelve el índice del elemento encontrado teniendo en cuenta la serie del array. En caso del fallo la función devuelve -1.

Nota

El elemento mínimo se busca tomando en cuenta el valor de la bandera AS_SERIES.

Las funciones ArrayMaximum y ArrayMinimum reciben como parámetro el array de cualquiera dimensión, la búsqueda se realiza sólo para la primera (cero) dimensión.

Ejemplo:

```
#property description "El indicador muestra las velas del mayor período en el período
//--- ajustes del indicador
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- plot 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- plot 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
```

© 2000-2025, MetaQuotes Ltd.

```
//---- plot 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- plot 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- plot 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- constante predefinida
#define INDICATOR_EMPTY_VALUE 0.0
//--- parámetros de entrada
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Período para el cálculo del
input datetime        InpDateStart=D'2013.01.01 00:00'; // Fecha de inicio del análisi
//--- búferes de indicadores para las velas bajistas
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- búferes de indicadores para las velas alcistas
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- variables globales
datetime ExtTimeBuff[];      // búfer de tiempo del período mayor
int      ExtSize=0;          // tamaño del búfer de tiempo
int      ExtCount=0;         // índice dentro del búfer de tiempo
int      ExtStartPos=0;      // posición inicial para el cálculo del indicador
bool     ExtStartFlag=true;  // bandera auxiliar para recibir la posición inicial
datetime ExtCurrentTime[1];  // última hora de formación de la barra desde el período
datetime ExtLastTime;        // última hora desde el período mayor para el que se ha h
bool     ExtBearFlag=true;   // bandera para determinar el orden de escritura de datos
bool     ExtBullFlag=true;   // bandera para determinar el orden de escritura de datos
int      ExtIndexMax=0;      // índice del elemento máximo del array
int      ExtIndexMin=0;      // índice del elemento mínimo del array
int      ExtDirectionFlag=0; // dirección de movimiento del precio para la vela actual
//--- espacio entre el precio de apertura y cierre de la vela para una correcta repres
```

© 2000-2025, MetaQuotes Ltd.

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//| Coloreado de la parte básica de la vela                          |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- buscamos los índices del elemento máximo y del mínimo en los arrays
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // el máximo en High
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // el mínimo en Low
//--- determinamos el número de barras desde el período actual que vamos a colear
   int count=fill_index-start+1;
//--- si el precio de cierre de la primera barra supera el precio de cierre de la últi
   if(open[start]>close[last])
     {
      //--- si hasta este momento la vela era bajista, entonces limpiamos los valores
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- vela bajista
      ExtDirectionFlag=-1;
      //--- formamos la vela
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- salida de la función
      return;
     }
//--- si el precio de cierre de la primera barra es inferior al precio de cierre de la
   if(open[start]<close[last])
     {
      //--- si hasta este momento la vela era alcistas, entonces limpiamos los valores
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- vela alcista
      ExtDirectionFlag=1;
      //--- formamos la vela
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- salida de la función
      return;
     }
//--- si se encuentra en esta parte de la función, el precio de apertura de la primera
//--- precio de cierre de la última barra; vamos a considerar esta vela bajista
//--- si hasta este momento la vela era bajista, entonces limpiamos los valores de los
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- vela bajista
   ExtDirectionFlag=-1;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si los precios de cierre y de apertura son iguales, utilizamos el desplazamiento
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//| Coloreado de la punta de la vela                                 |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- no dibujamos en caso de sólo una barra
   if(last-start==0)
      return;
//--- si el precio de cierre de la primera barra supera el precio de cierre de la últi
   if(open[start]>close[last])
     {
      //--- formamos la punta de la vela
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- salida de la función
      return;
     }
//--- si el precio de cierre de la primera barra es inferior al precio de cierre de la
   if(open[start]<close[last])
     {
      //--- formamos la punta de la vela
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- salida de la función
      return;
     }
//--- si se encuentra en esta parte de la función, el precio de apertura de la primera
//--- precio de cierre de la última barra; vamos a considerar esta vela bajista
//--- formamos la punta de la vela
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEm
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- chequeo del período de tiempo del indicador
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- visualización de los precios en el primer plano
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- enlace de los búferes de indicadores
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- fijamos algunos valores de propiedades para construir el indicador
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // tipo de construcción grá
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // estilo de la línea
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // grosor de la línea
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- si todavía no hay barras calculadas,
   if(prev_calculated==0)
     {
      //--- obtenemos la hora de aparición de las barras desde el período mayor
      if(!GetTimeData())
         return(0);
     }
//--- establecemos la dirección directa para la indexación
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- variable de inicio para el cálculo de las barras
   int start=prev_calculated;
//--- si la barra se está formando, volvemos a calcular el valor del indicador sobre e
   if(start!=0 && start==rates_total)
      start--;
//--- ciclo de cálculo de los valores del indicador
   for(int i=start;i<rates_total;i++)
     {
      //--- llenamos los elementos "i" de los búferes de indicadores con valores vacío
      FillIndicatorBuffers(i);
      //--- hacemos el cálculo para las barras a partir de la fecha InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- definimos por primera vez la posición a partir de la cual empezamos a r
         if(ExtStartFlag)
           {
            //--- recordamos el número de la barra inicial
            ExtStartPos=i;
            //--- determinamos la primera fecha desde el período mayor que supera time
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- cambiamos el valor de la bandera para no volver a entrar en este blo
            ExtStartFlag=false;
           }
         //--- comprobamos si hay más elementos en el array
         if(ExtCount<ExtSize)
           {
            //--- esperamos hasta que el valor de tiempo del período actual alcance el
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- dibujamos la parte básica de la vela (sin colorear entre la últir
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,Extl
               //--- coloreamos la punta de la vela (área entre la última barra y la p
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
```

© 2000-2025, MetaQuotes Ltd.

```
                //--- movemos la posición inicial para dibujar la siguiente barra
                ExtStartPos=i;
                //--- aumentamos el contador del array
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- anulamos los valores del error
             ResetLastError();
             //--- obtenemos la última fecha del período mayor
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Error del copiado de datos, código = ",GetLastError());
                return(0);
               }
             //--- si nueva fecha es mayor, terminamos la formación de la vela
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- limpiamos el área entre la última barra y la penúltima en los búf
                ClearEndOfBodyMain(i-1);
                //--- coloreamos esta área utilizando los búferes de indicadores auxili
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- movemos la posición inicial para dibujar la siguiente barra
                ExtStartPos=i;
                //--- reseteamos la bandera de dirección del precio
                ExtDirectionFlag=0;
                //--- recordamos la última fecha nueva
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- formamos la vela
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Prueba de la corrección del período del indicador introducido    |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- el período del indicador tiene que ser más grande que el período de tiempo (time
```

© 2000-2025, MetaQuotes Ltd.

```
   if(current_period>=high_period)
     {
      Print("¡Error! ¡El valor del período del indicador tiene que superar al valor de
      return(false);
     }
//--- si el período del indicador es una semana o un mes, entonces el período es corre
   if(high_period>32768)
      return(true);
//--- convertimos los valores de los períodos a los minutos
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- el período del indicador debe de ser múltiple del período de tiempo en el que se
   if(high_period%current_period!=0)
     {
      Print("¡Error! ¡El valor del período del indicador tiene que múltiple del valor
      return(false);
     }
//--- el período del indicador tiene que superar al período de tiempo (time frame) en
   if(high_period/current_period<3)
     {
      Print("¡Error! ¡El valor del período del indicador tiene que superar al valor de
      return(false);
     }
período del indicador es correcto para el período de tiempo actual
   return(true);
  }
//+------------------------------------------------------------------+
//| Recepción de datos de tiempo desde el período de tiempo mayor    |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- resetear el valor del error
   ResetLastError();
//--- copiamos todos los datos para la hora actual
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- obtenemos el código del error
      int code=GetLastError();
      //--- imprimimos el texto del error
      PrintFormat("¡Error al copiar datos! %s",code==4401
                  ? "¡El historial se sigue cargando!"
                  : "Código = "+IntegerToString(code));
      //--- devolvemos false para volver a intentar cargar datos
      return(false);
     }
//--- obtenemos el tamaño del array
   ExtSize=ArraySize(ExtTimeBuff);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ponemos el índice del ciclo para el array igual a cero
   ExtCount=0;
//--- ponemos la posición de la vela actual en este período de tiempo igual a cero
   ExtStartPos=0;
   ExtStartFlag=true;
//--- recordamos el último valor de tiempo desde el período de tiempo mayor
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- ejecución con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| La función forma la parte básica de la vela. Dependiendo del valor de|
//| la bandera, la función determina qué datos y en qué arrays deben   |
//| escribirse para una visualización correcta.                        |
//+------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- comprobamos el valor de la bandera
   if(flag)
     {
      //--- formamos el cuerpo de la vela
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- formamos la sombra de la vela
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- formamos el cuerpo de la vela
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- formamos la sombra de la vela
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------+
//| La función forma la punta de la vela. Dependiendo del valor de la bandera,  |
//| la función determina qué datos y en qué arrays deben             |
//| escribirse para una visualización correcta.                      |
//+------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- comprobamos el valor de la bandera
```

© 2000-2025, MetaQuotes Ltd.

```
   if(flag)
     {
      //--- formamos el fin del cuerpo de la vela
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- formamos el fin de la sombra de la vela
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- cambiamos el valor de la bandera al opuesto
      flag=false;
     }
   else
     {
      //--- formamos el fin del cuerpo de la vela
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- formamos el fin de la sombra de la vela
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- cambiamos el valor de la bandera al opuesto
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Limpiamos la punta de la vela (área entre la última barra y la penúltima     |
//| barra)                                                           |
//+------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------+
//| Limpieza de la vela                                              |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- prueba
   if(count!=0)
     {
      //--- llenamos los búferes indicadores con valores vacíos
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------+
//| Formación de la parte básica de la vela                          |
//+------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- prueba
   if(count!=0)
     {
      //--- llenamos los búferes indicadores con valores
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| Formación de la punta de la vela                                 |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- llenamos los búferes indicadores con valores
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Llenar los elementos "i" de los búferes de indicadores con valores vacíos|
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- establecemos un valor vacío en la cédula de búferes de indicadores
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayPrint

Muestra, en el registro, una matriz de tipo o estructura simple.

```
void  ArrayPrint(
   const void&   array[],             // matriz de salida
   uint          digits=_Digits,      // número de decimales después de la coma
   const string  separator=NULL,      // delimitador entre los valores de los campos d
   ulong         start=0,             // índice del primer elemento de salida
   ulong         count=WHOLE_ARRAY,   // número de elementos de salida
   ulong         flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_
   );
```

Parámetros

array[]

[in] Matriz de tipo o estructura simple.

digits=_Digits

[in] Número de decimales después de la coma para tipos reales. Por defecto, igual a _Digits.

separator=NULL

[in] Delimitador entre los valores de los campos del elemento de la estructura. El valor por defecto NULL indica una cadena vacía, en este caso, el delimitador es un espacio.

start=0

[in] Índice del primer elemento de salida de la matriz. Por defecto, es mostrado desde el índice cero.

count=WHOLE_ARRAY

[in] Número de elementos de la matriz que se necesitan mostrar. Por defecto, es mostrada toda la matriz (count=WHOLE_ARRAY).

flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_ALIGN

[in] Combinación de marcas especificadas en el modo de salida. Por defecto, todas las marcas:

- ARRAYPRINT_HEADER - visualización de encabezados para la matriz de las estructuras
- ARRAYPRINT_INDEX - visualización a la izquierda del número de índice
- ARRAYPRINT_LIMIT - visualización solo de los 100 primeros y 100 últimos elementos de la matriz. Se utiliza cuando se desea mostrar sólo una parte de una matriz más amplia.
- ARRAYPRINT_ALIGN - permitir la alineación de los valores de salida, es decir, los números estarán alineados a la derecha, mientras que las líneas, a la izquierda.
- ARRAYPRINT_DATE - al visualizar el datetime, muestra la fecha en formato dd.mm.yyyy
- ARRAYPRINT_MINUTES - al visualizar el datetime, muestra la fecha en formato HH:MM
- ARRAYPRINT_SECONDS - al visualizar el datetime, muestra la fecha en formato HH:MM:SS

Valor de retorno

No

Nota

ArrayPrint() muestra, en el registro, no todos los campos de la matriz de estructuras, ya que tanto los campos de matrices como los de punteros a objetos son omitidos. Estas columnas simplemente

© 2000-2025, MetaQuotes Ltd.

no serán expuestas para imprimirse, esto es así con el fin de tener una presentación simple y conveniente. Si necesita la visualización de todos los campos de esta estructura, entonces usted necesita escribir su propia función de salida en masa con el formato deseado.

Ejemplo:

```
//--- muestra los valores de las últimas 10 barras
   MqlRates rates[];
   if(CopyRates(_Symbol,_Period,1,10,rates))
     {
      ArrayPrint(rates);
      Print("Inspección\n[time]\t[open]\t[high]\t[low]\t[close]\t[tick_volume]\t[sprea
      for(int i=0;i<10;i++)
        {
         PrintFormat("[%d]\t%s\t%G\t%G\t%G\t%G\t%G\t%G\t%I64d\t",i,
         TimeToString(rates[i].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS),
         rates[i].open,rates[i].high,rates[i].low,rates[i].close,
         rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
   else
      PrintFormat("CopyRates failed, error code=%d",GetLastError());
//--- ejemplo de salida
/*
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295         18110       10   17
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.11930 1.12747         17829        9   15
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744         13458       10    9
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194         15362        9   12
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172         16833        9   12
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052         15933        8   10
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528         11888        9    8
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915          7284       10    5
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904          8710        9    6
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263          8956        7    7
   Inspección
   [time] [open] [high] [low] [close] [tick_volume] [spread] [real_volume]
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295 18110 10 17300175000
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.1193 1.12747 17829 9 15632176000
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744 13458 10 9593492000
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194 15362 9 12352245000
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172 16833 9 12961333000
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052 15933 8 10720384000
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528 11888 9 8084811000
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915 7284 10 5087113000
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904 8710 9 6769629000
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263 8956 7 7192138000
*/
```

Ver también

© 2000-2025, MetaQuotes Ltd.

FileSave, FileLoad

© 2000-2025, MetaQuotes Ltd.

ArrayRange

Esta función devuelve el número de elementos en la dimensión especificada del array.

```
int  ArrayRange(
   const void&   array[],      // array a comprobar
   int           rank_index    // número de dimensión
   );
```

Parámetros

array[]

[in] Array a comprobar.

rank_index

[in] Índice de dimensión.

Valor devuelto

Número de elementos en la dimensión especificada del array.

Nota

Puesto que los índices se empiezan desde cero, el número de dimensiones del array es a uno más grande que el índice de la última dimensión.

Ejemplo:

```
void OnStart()
  {
//--- creamos un array de 4 dimensiones
   double array[][5][2][4];
//--- fijamos el tamaño de la dimensión-0
   ArrayResize(array,10,10);
//--- imprimimos dimensiones
   int temp;
   for(int i=0;i<4;i++)
     {
      //--- obtenemos el tamaño de dimensión i
      temp=ArrayRange(array,i);
      //--- imprimimos
      PrintFormat("dim = %d, range = %d",i,temp);
     }
//--- Resultado
// dim = 0, range = 10
// dim = 1, range = 5
// dim = 2, range = 2
// dim = 3, range = 4
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayResize

Esta función establece nuevo tamaño en la primera dimensión del array.

```
int  ArrayResize(
   void&  array[],              // array pasado por referencia
   int    new_size,             // nuevo tamaño del array
   int    reserve_size=0        // valor de reserva del tamaño (sobrante)
   );
```

Parámetros

array[]

[out] Array para el cambio de tamaño.

new_size

[in] Nuevo tamaño para la primera dimensión.

reserve_size=0

[in] Tamaño para la reserva adicional.

Valor devuelto

Si se ejecuta con éxito, la función devuelve la cantidad de todos los elementos contenidos en el array después del cambio de tamaño; de lo contrario devuelve -1 y el array no cambia sus dimensiones.

Si ArrayResize() se aplica a una matriz estática, a una serie temporal o un búfer de indicador, el tamaño de la matriz permanerecerá igual: estas matrices no pueden ser redistribuidas. En este caso, si *new_size*<=ArraySize(*array*), la función simplemente retornará *new_size*; en caso contrario, retornará -1.

Nota

Esta función puede aplicarse sólo a los arrays dinámicos. Además, hay que tener en cuenta que no se puede cambiar el tamaño de los arrays dinámicos que han sido asignados como búfers de indicadores por la función SetIndexBuffer(). Todas las operaciones relacionadas con el cambio del tamaño de los búfers de indicadores se realizan por el subsistema ejecutivo del terminal.

El número total de los elementos del array no puede superar 2147483647.

En caso de la distribución frecuente de la memoria se recomienda utilizar el tercer parámetro que establece una reserva para disminuir la cantidad de distribución física de la memoria. Las siguientes llamadas a la función ArrayResize no llevan a la redistribución física de la memoria, simplemente se cambia el tamaño de la primera dimensión del array dentro de los límites de la memoria reservada. Hay que recordar que el tercer parámetro va a utilizarse sólo cuando va a tener lugar la distribución física de la memoria, por ejemplo:

```
ArrayResize(arr,1000,1000);
for(int i=1;i<3000;i++)
   ArrayResize(arr,i,1000);
```

En este caso se producirá 2 redistribuciones de la memoria: una vez antes de la entrada en el ciclo de 3000 elementos, en este caso la dimensionalidad del array se establecerá en 1000, y la segunda -

© 2000-2025, MetaQuotes Ltd.

cuando i es igual a 2000. Si omitimos el tercer parámetro, habrá 2000 redistribuciones físicas de la memoria, y esto ralentizará la ejecución del programa.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- contadores
   ulong start=GetTickCount();
   ulong now;
   int   count=0;
//--- array para demostración de la versión rápida
   double arr[];
   ArrayResize(arr,100000,100000);
//--- comprobamos con qué rapidez funciona la versión con la reserva de memoria
   Print("--- Test Fast: ArrayResize(arr,100000,100000)");
   for(int i=1;i<=300000;i++)
     {
      //--- ¡fijamos el nuevo tamaño del array indicando la reserva de 100000 elemento
      ArrayResize(arr,i,100000);
      //--- al alcanzar un número redondo, mostramos el tamaño del array y el tiempo q
      if(ArraySize(arr)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(arr)=%d Time=%d ms",count,ArraySize(arr),(now-star
         start=now;
        }
     }
//--- ahora mostramos qué lento trabaja la versión sin la reserva de la memoria
   double slow[];
   ArrayResize(slow,100000,100000);
//---
   count=0;
   start=GetTickCount();
   Print("---- Test Slow: ArrayResize(slow,100000)");
//---
   for(int i=1;i<=300000;i++)
     {
      //--- fijamos el nuevo tamaño del array, pero ya sin la reserva adicional
      ArrayResize(slow,i);
      //--- al alcanzar un número redondo, mostramos el tamaño del array y el tiempo q
      if(ArraySize(slow)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(slow)=%d Time=%d ms",count,ArraySize(slow),(now-st
         start=now;
        }
     }
  }
//--- Un resultado aproximado de la ejecución del script
/*
   Test_ArrayResize (EURUSD,H1)   --- Test Fast: ArrayResize(arr,100000,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(arr)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(arr)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(arr)=300000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   ---- Test Slow: ArrayResize(slow,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(slow)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(slow)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(slow)=300000 Time=228511 ms
*/
```

© 2000-2025, MetaQuotes Ltd.

Véase también

ArrayInitialize

© 2000-2025, MetaQuotes Ltd.

ArrayInsert

Inserta en la matriz-receptor el número indicado de elementos, comenzando por el índice establecido.

```
bool  ArrayInsert(
   void&        dst_array[],          // matriz-receptor
   const void&  src_array[],          // matriz-fuente
   uint         dst_start,            // índice de la matriz-receptor para la inserci
   uint         src_start=0,          // índice en la matriz-fuente para el copiado
   uint         count=WHOLE_ARRAY     // número de elementos a insertar
   );
```

Parámetros

dst_array[]

[in][out] Matriz-receptor a la que se deben añadir los elementos.

src_array[]

[in] Matriz-fuenete desde la que se deben añadir los elementos.

dst_start

[in] Índice en la matriz-receptor para la inserción de los elementos de la matriz-fuente.

src_start=0

[in] Índice de la matriz-fuente, a partir del cual se toman los elementos de la matriz-fuente para la inserción.

count=WHOLE_ARRAY

[in] Número de elementos a añadir desde la matriz-fuente. El valor WHOLE_ARRAY indica que se insertarán todos los elementos desde el índice establecido hasta el final de la matriz.

Valor retornado

Retorna true en caso de éxito, de lo contrario, false. Para obtener información sobre el error, es necesario llamar la función GetLastError(). Posibles errores:

- 5052 - ERR_SMALL_ARRAY (los parámetros *start* y/o *count* se han indicado incorrectamente o la matriz-fuente *src_array[]* está vacía),
- 5056 - ERR_SERIES_ARRAY (la matriz no puede modificarse, búfer de indicador),
- 4006 - ERR_INVALID_ARRAY (el copiado en sí no está permitido, o bien las matrices tienen tipo distinto, o bien una matriz de tamaño fijo que contiene objetos de clase o estructuras con destructores),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (la matriz no contiene estructuras POD, es decir, el copiado simple no es posible),
- errores de cambio de tamaño de la matriz-receptor *dst_array[]* - estos se muestran en la descripción de la función ArrayRemove().

Observación

Si la función se usa para una matriz de tamaño fijo, el propio tamaño de la matriz-receptor *dst_array[]* no cambia, en este caso, además, comenzando desde la posición *dst_start*, los

© 2000-2025, MetaQuotes Ltd.

elementos de la matriz-receptor se desplazan a la derecha (los últimos elementos *count* "se caen"), y en el lugar liberado tiene lugar el copiado de elementos de la matriz-fuente.

No se puede insertar elementos en matrices dinámicas designadas como búferes de indicador con la función SetIndexBuffer(). Para los búferes de indicador, todas las operaciones de cambio de tamaño son realizadas por el subsistema ejecutor del terminal.

En la matriz-fuente se copian los elementos, comenzando por el índice *src_start*. El tamaño de la matriz-fuente, en este caso, no cambia. Los elementos añadidos a la matriz-receptor no son enlaces a los elementos de la matriz-fuente, esto significa que los posteriores cambios de los elementos en cualquiera de las dos matrices no se reflejarán en la segunda.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos una matriz de tamaño fijo y rellenamos con valores
   int array_dest[10];
   for(int i=0;i<10;i++)
     {
      array_dest[i]=i;
     }
   //--- matriz-fuente
   int array_source[10];
   for(int i=0;i<10;i++)
     {
      array_source[i]=10+i;
     }
//--- mostramos las matrices antes de la inserción de los elementos
   Print("Antes de llamar ArrayInsert()");
   ArrayPrint(array_dest);
   ArrayPrint(array_source);
//--- insertamos 3 elementos de la matriz-fuente y mostramos la nueva composición de l
   ArrayInsert(array_dest,array_source,4,0,3);
   Print("Después de llamar ArrayInsert()");
   ArrayPrint(array_dest);
/*
  Resultado de la ejecución
   Antes de llamar ArrayInsert()
   0 1 2 3 4 5 6 7 8 9
   Después de llamar ArrayInsert()
   0 1 2 3 10 11 12 7 8 9
*/
```

Mire también

ArrayRemove, ArrayCopy, ArrayResize, ArrayFree

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

ArrayRemove

Elimina de la matriz el número indicado de elementos, comenzando por el índice indicado.

```
bool  ArrayRemove(
   void&         array[],            // matriz de cualquier tipo
   uint          start,              // desde qué índice comenzamos a eliminar
   uint          count=WHOLE_ARRAY   // número de elementos
   );
```

Parámetros

array[]

[in][out] Matriz.

start

[in] Índice desde el que se comienza a eliminar los elementos de la matriz.

count=WHOLE_ARRAY

[in] Número de elementos a eliminar. El valor WHOLE_ARRAY indica la eliminación de todos los elementos desde el índice establecido hasta el final de la matriz.

Valor retornado

Retorna true en caso de éxito, de lo contrario, false. Para obtener información sobre el error, es necesario llamar la función GetLastError(). Posibles errores:

- 5052 - ERR_SMALL_ARRAY (el valor *start* es demasiado grande),
- 5056 - ERR_SERIES_ARRAY (la matriz no puede modificarse, búfer de indicador),
- 4003 - ERR_INVALID_PARAMETER (el valor *count* es demasiado grande),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (matriz de tamaño fijo, que contiene objetos complejos con destructor),
- 4006 - ERR_INVALID_ARRAY (matriz de tamaño fijo, que contiene objetos de estructuras o clases con destructor).

Observación

Si la función se usa para una matriz de tamaño fijo, el propio tamaño de la matriz no cambia: en este caso, además, tiene lugar el copiado físico de la "cola" restante en la posición *start*. Para comprender exactamente el funcionamiento de la función, mire el ejemplo de abajo. El copiado "físico" significa que los objetos copiados no se crean con la ayuda de la llamada de un constructor u operador de copiado, sino que simplemente sucede el copiado de la representación binaria del objeto. Precisamente por este motivo se prohíbe aplicar la función ArrayRemove() a una matriz de tamaño fijo que contenga objetos con destructor (se mostrará el error ERR_INVALID_ARRAY o ERR_STRUCT_WITHOBJECTS_ORCLASS). Así, al eliminar este objeto el destructor deberá ser llamado dos veces: para el objeto original y para su copia.

No es posible eliminar los elementos de las matrices dinámicas designadas como búferes de indicador con la función SetIndexBuffer(), esto provocará la aparición del error ERR_SERIES_ARRAY. Para los búferes de indicador, todas las operaciones de cambio de tamaño son realizadas por el subsistema ejecutor del terminal.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos una matriz de tamaño fijo y rellenamos con valores
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- mostramos la matriz antes de eliminar los elementos
   Print("Antes de llamar ArrayRemove()");
   ArrayPrint(array);
//--- eliminamos 2 elementos de la matriz y mostramos la nueva composición
   ArrayRemove(array,4,2);
   Print("Después de llamar ArrayRemove()");
   ArrayPrint(array);
/*
  Resultado de la ejecución:
  Antes de llamar ArrayRemove()
   0 1 2 3 4 5 6 7 8 9
  Después de llamar ArrayRemove()
   0 1 2 3 6 7 8 9 8 9
*/
```

Mire también

ArrayInsert, ArrayCopy, ArrayResize, ArrayFree

© 2000-2025, MetaQuotes Ltd.

ArrayReverse

Invierte en la matriz el número indicado de elementos, comenzando por el índice indicado.

```
bool  ArrayReverse(
   void&         array[],            // matriz de cualquier tipo
   uint          start=0,            // desde qué índice comenzamos a invertir la matri
   uint          count=WHOLE_ARRAY   // número de elementos
   );
```

Parámetros

array[]

[in][out] Matriz.

start=0

[in] Índice desde el que comienza a invertirse la matriz.

count=WHOLE_ARRAY

[in] Número de elementos invertidos. Si se indica el valor WHOLE_ARRAY, se desplazarán a la inversa entre sí todos los elementos de la matriz, comenzando por el índice establecido *start* y hasta el final de la matriz.

Valor retornado

Retorna true en caso de éxito, de lo contrario, false.

Observación

La función ArraySetAsSeries() no desplaza físicamente los elementos de la matriz, sino que solo invierte la dirección para organizar el acceso a los elementos como en la serie temporal. La matriz ArrayReverse() desplaza físicamente los elementos, de tal forma que la matriz "se invierte".

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos una matriz de tamaño fijo y rellenamos con valores
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- mostramos la matriz antes de invertir los elementos
   Print("Antes de llamar ArrayReverse()");
   ArrayPrint(array);
//--- invertimos 3 elementos en la matriz y mostramos la nueva composición
   ArrayReverse(array,4,3);
   Print("Después de llamar ArrayReverse()");
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayPrint(array);
/*
  Resultado de la ejecución:
  Antes de llamar ArrayReverse()
   0 1 2 3 4 5 6 7 8 9
  Después de llamar ArrayReverse()
   0 1 2 3 6 5 4 7 8 9
*/
```

Mire también

ArrayInsert, ArrayRemove, ArrayCopy, ArrayResize, ArrayFree, ArrayGetAsSeries, ArraySetAsSeries

© 2000-2025, MetaQuotes Ltd.

ArraySetAsSeries

Esta función pone la bandera AS_SERIES al objeto del array dinámico especificado, la indexación de los elementos del array va a efectuarse como en las series temporales.

```
bool  ArraySetAsSeries(
   const void&  array[],     // array por referencia
   bool         flag         // true significa el orden inverso de indexación
   );
```

Parámetros

array[]

[in][out] Array numérico para la puesta.

flag

[in] Dirección de indexación del array.

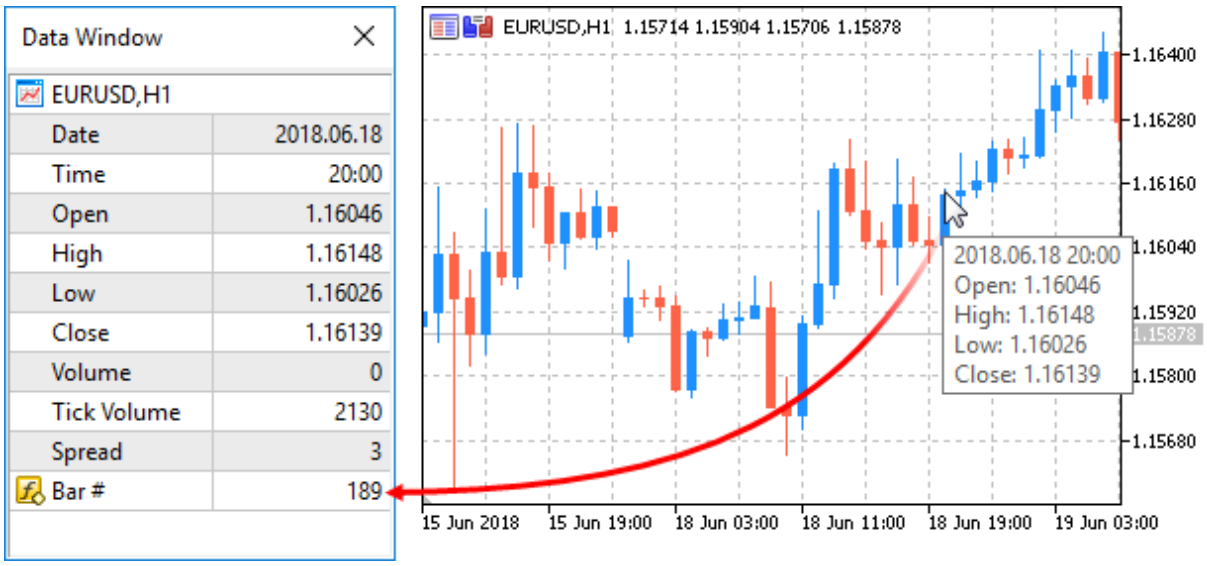
Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

No se puede poner la bandera AS_SERIES para los arrays multidimensionales y estáticos (es decir, los arrays cuyo tamaño ya está indicado en los corchetes en la etapa de compilación). La indexación en una serie temporal se diferencia de un array usual en que la indexación de los elementos de la serie temporal se realiza del fin del array al inicio (de los datos más recientes a los más antiguos).

Ejemplo: indicador que muestra el número de la barra



© 2000-2025, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Numeration
#property indicator_label1  "Numeration"
#property indicator_type1   DRAW_LINE
#property indicator_color1  CLR_NONE
//--- indicator buffers
double         NumerationBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- establecemos la indexación para el buffer como en serie temporal
   ArraySetAsSeries(NumerationBuffer,true);
//--- establecemos la precisión de representación en DataWindow
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- como va a visualizarse el nombre del array de indicador en DataWindow
   PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---  vamos a guardar la hora de apertura de la barra cero actual
   static datetime currentBarTimeOpen=0;
//--- revertimos el acceso al array time[] - lo hacemos como en serie temporal
   ArraySetAsSeries(time,true);
//--- Si la hora de barra cero se diferencia de la que estamos guardando,
   if(currentBarTimeOpen!=time[0])
     {
     //--- vamos a enumerar todos las barras desde el momento actual hacia dentro del
      for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
      currentBarTimeOpen=time[0];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Acceso a las series temporales, ArrayGetAsSeries

© 2000-2025, MetaQuotes Ltd.

ArraySize

Esta función devuelve el número de elementos del array especificado.

```
int  ArraySize(
   const void&  array[]    // array a comprobar
   );
```

Parámetros

array[]

[in] Array de cualquier tipo.

Valor devuelto

Valor del tipo int.

Nota

Para un array unidimensional el valor devuelto por la función ArraySize es igual al valor de ArrayRange(array,0).

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- creación de arrays
   double one_dim[];
   double four_dim[][10][5][2];
//--- tamaños
   int one_dim_size=25;
   int reserve=20;
   int four_dim_size=5;
//--- variable auxiliar
   int size;
//--- adjudicamos memoria sin el backup
   ArrayResize(one_dim,one_dim_size);
   ArrayResize(four_dim,four_dim_size);
//--- 1. array unidimensional
   Print("+============================================================+");
   Print("Tamaños de arrays:");
   Print("1. Array unidimensional");
   size=ArraySize(one_dim);
   PrintFormat("Tamaño de la dimensión cero = %d, Tamaño del array = %d",one_dim_size,
//--- 2. array multidimensional
   Print("2. Array multidimensional");
   size=ArraySize(four_dim);
   PrintFormat("Tamaño de la dimensión cero = %d, Tamaño del array = %d",four_dim_size
//--- tamaños de dimensiones
   int d_1=ArrayRange(four_dim,1);
   int d_2=ArrayRange(four_dim,2);
   int d_3=ArrayRange(four_dim,3);
   Print("Prueba:");
   Print("Dimensión cero = Tamaño del array / (Primera dimensión * Segunda dimensión *
   PrintFormat("%d = %d / (%d * %d * %d)",size/(d_1*d_2*d_3),size,d_1,d_2,d_3);
//--- 3. array unidimensional con memoria backup
   Print("3. Array unidimensional con memoria backup");
//--- aumentamos el valor 2 veces
   one_dim_size*=2;
//--- adjudicamos la memoria con backup
   ArrayResize(one_dim,one_dim_size,reserve);
//--- imprimimos el tamaño
   size=ArraySize(one_dim);
   PrintFormat("Tamaño con backup = %d, Tamaño real del array = %d",one_dim_size+reser
  }
```

© 2000-2025, MetaQuotes Ltd.

ArraySort

Ordena el array numérico multidimensional por orden ascendente de los valores en la primera dimensión.

```
bool  ArraySort(
   void&  array[]      // array para ordenar
   );
```

Parámetros

array[]

[in][out] Array numérico para ser ordenado.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

El array siempre se ordena en orden ascendente independientemente del valor de la bandera AS_SERIES.

Las funciones ArraySort y ArrayBSearch reciben como parámetro el array de cualquiera dimensión, la ordenación y la búsqueda se realiza sólo para la primera (cero) dimensión.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#property description "El indicador analiza los datos del último mes y colorea todas l
#property description "volúmenes de ticks grandes y pequeños. Para determinar estas ve
#property description "array de los volúmenes de ticks. Las velas cuyos volúmenes se c
#property description "por cientos del array se consideran pequeñas. Las velas cuyos v
#property description "los últimos InpBigVolume por cientos del array se consideran gr
//--- ajustes del indicador
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot
#property indicator_label1  "VolumeFactor"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrDodgerBlue,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- constante predefinida
#define INDICATOR_EMPTY_VALUE 0.0
//--- parámetros de entrada
input int InpSmallVolume=15; // Porcentaje de volúmenes pequeños (<50)
input int InpBigVolume=20;   // Porcentaje de volúmenes grandes (<50)
//--- hora de inicio de análisis (va a desplazarse)
datetime ExtStartTime;
//--- búferes de indicadores
double   ExtOpenBuff[];
double   ExtHighBuff[];
double   ExtLowBuff[];
double   ExtCloseBuff[];
double   ExtColorBuff[];
//--- valores límite de volúmenes para visualización de velas
long     ExtLeftBorder=0;
long     ExtRightBorder=0;
//+------------------------------------------------------------------+
//| Recepción de valores de los límites para volúmenes de ticks      |
//+------------------------------------------------------------------+
bool GetVolumeBorders(void)
  {
//--- variables
   datetime stop_time;  // hora del fin de copiado
   long     buff[];     // búfer al que vamos a copiar
//--- hora final - hora actual
   stop_time=TimeCurrent();
//--- hora de inicio - un mes antes del tiempo actual
   ExtStartTime=GetStartTime(stop_time);
//--- obtenemos los valores de volúmenes de ticks
   ResetLastError();
   if(CopyTickVolume(Symbol(),Period(),ExtStartTime,stop_time,buff)==-1)
     {
      //--- no ha salido conseguir datos, devolvemos false para iniciar el comando de
      PrintFormat("No se ha podido conseguir los valores del volumen de ticks. Código
      return(false);
     }
//--- calculamos el tamaño del array
   int size=ArraySize(buff);
//--- clasificamos array
   ArraySort(buff);
//--- determinamos los valores del límite izquierdo y derecho para los volúmenes de ti
   ExtLeftBorder=buff[size*InpSmallVolume/100];
   ExtRightBorder=buff[(size-1)*(100-InpBigVolume)/100];
//--- ejecución con éxito
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Recepción de la fecha que sea un mes más antigua de la fecha que ha sido pasada
//+------------------------------------------------------------------+
datetime GetStartTime(const datetime stop_time)
  {
//--- convertimos el tiempo de finalización a la variable de la estructura del tipo Mq
   MqlDateTime temp;
   TimeToStruct(stop_time,temp);
//--- obtenemos la fecha que sea un mes más antigua
   if(temp.mon>1)
      temp.mon-=1;  // el mes en curso no es el primero de este año, entonces el numer
   else
     {
      temp.mon=12;  // el mes en curso es el primer mes del año, entonces el número de
      temp.year-=1; // y el número del año será menos uno
     }
//--- el número del día no va a superar 28
   if(temp.day>28)
      temp.day=28;
//--- devolvemos la fecha obtenida
   return(StructToTime(temp));
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- comprobar si los parámetros de entrada satisfacen las condiciones
   if(InpSmallVolume<0 || InpSmallVolume>=50 || InpBigVolume<0 || InpBigVolume>=50)
     {
      Print("Parámetros de entrada incorrectos");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,ExtOpenBuff);
   SetIndexBuffer(1,ExtHighBuff);
   SetIndexBuffer(2,ExtLowBuff);
   SetIndexBuffer(3,ExtCloseBuff);
   SetIndexBuffer(4,ExtColorBuff,INDICATOR_COLOR_INDEX);
//--- establecemos el valor que no va a mostrarse
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- establecemos las etiquetas para los búferes de indicadores
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- comprobamos si hay más barras sin procesar
```

© 2000-2025, MetaQuotes Ltd.

```
   if(prev_calculated<rates_total)
     {
      //--- obtenemos nuevos valores de los límites izquierdo y derecho para los volúr
      if(!GetVolumeBorders())
         return(0);
     }
//--- variable de inicio para el cálculo de barras
   int start=prev_calculated;
//--- si los valores del indicador ya han sido calculados en el tick anterior, trabaja
   if(start>0)
      start--;
//--- establecemos la dirección directa de indexación en series temporales
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
   ArraySetAsSeries(tick_volume,false);
//--- ciclo del cálculo de los valores del indicador
   for(int i=start;i<rates_total;i++)
     {
      //--- coloreamos las velas empezando de la fecha inicial
      if(ExtStartTime<=time[i])
        {
         //--- si el valor no es más bajo que el límite derecho, coloreamos la vela
         if(tick_volume[i]>=ExtRightBorder)
           {
            //--- obtenemos los datos para dibujar la vela
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- color DodgerBlue
            ExtColorBuff[i]=0;
            //--- seguimos con el ciclo
            continue;
           }
         //--- si el valor no es más alto que el límite izquierdo, coloreamos la vela
         if(tick_volume[i]<=ExtLeftBorder)
           {
            //--- obtenemos los datos para dibujar la vela
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- color Orange
            ExtColorBuff[i]=1;
            //--- seguimos con el ciclo
            continue;
           }
        }
      //--- para las barras que no han entrado en los cálculos, ponemos el valor vacío
      ExtOpenBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtHighBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtLowBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtCloseBuff[i]=INDICATOR_EMPTY_VALUE;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

Véase también

ArrayBsearch

© 2000-2025, MetaQuotes Ltd.

ArraySwap

Intercambia el contenido de dos matrices dinámicas del mismo tipo. Para las matrices multidimensionales, deberá coincidir el número de elementos en todas las dimensiones, excepto la primera.

```
bool  ArraySwap(
   void&  array1[],      // primera matriz
   void&  array2[]       // segunda matriz
   );
```

Parámetros

array1[]

[in][out] Matriz de tipo numérico.

array2[]

[in][out] Matriz de tipo numérico.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false. En este caso, GetLastError() retornará el código de error ERR_INVALID_ARRAY.

Nota

La función acepta matrices dinámicas del mismo tipo y de las mismas dimensiones, excepto la primera. Para los tipos enteros, el signo se ignora, es decir, char==uchar)

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- matrices para guardar las cotizaciones
   double source_array[][8];
   double   dest_array[][8];
   MqlRates rates[];
//--- obtenemos los datos de las últimas 20 velas en el marco temporal actual
   int copied=CopyRates(NULL,0,0,20,rates);
   if(copied<=0)
     {
      PrintFormat("CopyRates(%s,0,0,20,rates) failed, error=%d",
                  Symbol(),GetLastError());
      return;
     }
//--- establecemos el tamaño de la matriz conforme al número de datos copiados
   ArrayResize(source_array,copied);
//--- rellanamos la matriz rate_array_1[] con los datos de rates[]
   for(int i=0;i<copied;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       source_array[i][0]=(double)rates[i].time;
       source_array[i][1]=rates[i].open;
       source_array[i][2]=rates[i].high;
       source_array[i][3]=rates[i].low;
       source_array[i][4]=rates[i].close;
       source_array[i][5]=(double)rates[i].tick_volume;
       source_array[i][6]=(double)rates[i].spread;
       source_array[i][7]=(double)rates[i].real_volume;
      }
//--- realizamos el intercambio de datos entre source_array[] y dest_array[]
   if(!ArraySwap(source_array,dest_array))
     {
      PrintFormat("ArraySwap(source_array,rate_array_2) failed, error code=%d",GetLast
      return;
     }
//--- nos aseguramos de que tras el intercambio, el tamaño de la matriz-fuente se haya
   PrintFormat("ArraySwap() done: ArraySize(source_array)=%d",ArraySize(source_array))
//--- mostramos los datos de la matriz-receptor dest_array[]
   ArrayPrint(dest_array);
  }
```

Vea también

ArrayCopy, ArrayFill, ArrayRange, ArrayIsDynamic

© 2000-2025, MetaQuotes Ltd.

ArrayToFP16

Permite copiar un array de tipo float o double en un array de tipo ushort con el formato establecido.

```
bool   ArrayToFP16(
   const ushort&         dst_array[],         // a dónde copiamos
   const float&          src_array[],         // de dónde copiamos
   ENUM_FLOAT16_FORMAT   fmt                  // formato
   );
```

Sobrecarga para el tipo double

```
bool   ArrayToFP16(
   const ushort&         dst_array[],         // a dónde copiamos
   const double&         src_array[],         // de dónde copiamos
   ENUM_FLOAT16_FORMAT   fmt                  // formato
   );
```

Parámetros

dst_array[]

[out] Array receptor de tipo ushort.

src_array[]

[in] Array fuente de tipo float o double.

fmt

[in] Formato de copiado de la enumeración ENUM_FLOAT16_FORMAT.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

Los formatos FLOAT16 y BFLOAT16 se definen en la enumeración ENUM_FLOAT16_FORMAT y se usan en MQL5 sola y exclusivamente para trabajar con modelos ONNX.

La función se utiliza para convertir parámetros de entrada de tipo float o doulbe a tipo FLOAT16 y BFLOAT16. Estos parámetros de entrada se utilizan en la función OnnxRun.

FLOAT16, también conocido como formato de media precisión o "half-precision float", utiliza 16 bits para representar un número de coma flotante. Este formato ofrece un equilibrio entre precisión y eficiencia computacional. FLOAT16 se utiliza ampliamente en el aprendizaje profundo y las redes neuronales, donde se requiere un alto rendimiento al procesar grandes cantidades de datos. Este formato permite acelerar los cálculos reduciendo el tamaño de los números, lo cual resulta especialmente importante al entrenar redes neuronales profundas en unidades de procesamiento gráfico (GPU).

BFLOAT16 (o Brain Floating Point 16) también usa 16 bits, pero se diferencia de FLOAT16 en el modo de representación de los números. En este formato, se asignan 8 bits para representar el

© 2000-2025, MetaQuotes Ltd.

exponente y los 7 bits restantes se utilizan para representar la mantisa. Este formato se ha desarrollado para su uso en el aprendizaje profundo y la inteligencia artificial, especialmente en los procesadores Tensor Processing Unit (TPU) de Google. BFLOAT16 muestra un buen rendimiento en el entrenamiento de redes neuronales y puede utilizarse eficazmente para acelerar el cálculo.

Ejemplo: función del artículo Trabajamos con modelos ONNX en formatos float16 y float8

```
//+------------------------------------------------------------------+
//| RunCastFloat16ToDouble                                           |
//+------------------------------------------------------------------+
bool RunCastFloat16ToDouble(long model_handle)
  {
   PrintFormat("test=%s",__FUNCTION__);
   double test_data[12]= {1,2,3,4,5,6,7,8,9,10,11,12};
   ushort data_uint16[12];
   if(!ArrayToFP16(data_uint16,test_data,FLOAT_FP16))
     {
      Print("error in ArrayToFP16. error code=",GetLastError());
      return(false);
     }
   Print("test array:");
   ArrayPrint(test_data);
   Print("ArrayToFP16:");
   ArrayPrint(data_uint16);
   U<ushort> input_float16_values[3*4];
   U<double> output_double_values[3*4];
   float test_data_float[];
   if(!ArrayFromFP16(test_data_float,data_uint16,FLOAT_FP16))
     {
      Print("error in ArrayFromFP16. error code=",GetLastError());
      return(false);
     }
   for(int i=0; i<12; i++)
     {
      input_float16_values[i].value=data_uint16[i];
      PrintFormat("%d input value =%f  Hex float16 = %s  ushort value=%d",i,test_data_
     }
   Print("ONNX input array:");
   ArrayPrint(input_float16_values);
   bool res=OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float16_values,output_double
   if(!res)
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      return(false);
     }
   Print("ONNX output array:");
   ArrayPrint(output_double_values);
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   double sum_error=0.0;
   for(int i=0; i<12; i++)
     {
      double delta=test_data[i]-output_double_values[i].value;
      sum_error+=MathAbs(delta);
      PrintFormat("%d output double %f = %s  difference=%f",i,output_double_values[i].
     }
//---
   PrintFormat("test=%s   sum_error=%f",__FUNCTION__,sum_error);
//---
   return(true);
  }
```

Ver también

ArrayFromFP16, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

ArrayToFP8

Permite copiar un array de tipo float o double en un array de tipo uchar con el formato establecido.

```
bool   ArrayToFP8(
   const uchar&          dst_array[],        // a dónde copiamos
   const float&          src_array[],        // de dónde copiamos
   ENUM_FLOAT8_FORMAT    fmt                 // formato
   );
```

Sobrecarga para el tipo double

```
bool   ArrayToFP8(
   const uchar&          dst_array[],        // a dónde copiamos
   const double&         src_array[],        // de dónde copiamos
   ENUM_FLOAT8_FORMAT    fmt                 // formato
   );
```

Parámetros

dst_array[]

[out] Array receptor de tipo uchar.

src_array[]

[in] Array fuente de tipo float o double.

fmt

[in] Formato de copiado de la enumeración ENUM_FLOAT8_FORMAT.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

Todos los tipos de formato FP8 se definen en la enumeración ENUM_FLOAT8_FORMAT y se usan en MQL5 sola y exclusivamente para trabajar con modelos ONNX.

La función se utiliza para convertir parámetros de entrada de tipo float o double a uno de los tipos FP8. Estos parámetros de entrada se utilizan en la función OnnxRun.

El formato FP8 (coma flotante de 8 bits) es uno de los tipos de datos usados para representar números en coma flotante. En FP8, cada número se representa con 8 bits de datos, que suelen dividirse en tres componentes: signo, exponente y mantisa. Este formato ofrece un compromiso entre precisión y eficiencia de almacenamiento, lo cual lo hace atractivo para su uso en aplicaciones en las que es necesario ahorrar memoria y recursos computacionales.

Con su representación compacta de los números, el FP8 puede reducir los requisitos de memoria y acelerar los cálculos. Además, el FP8 puede ser útil para implementar operaciones de bajo nivel, como cálculos aritméticos y procesamiento de señales.

© 2000-2025, MetaQuotes Ltd.

Ejemplo: función del artículo Trabajamos con modelos ONNX en formatos float16 y float8

```
//+------------------------------------------------------------------+
//| RunCastFloat8Float                                               |
//+------------------------------------------------------------------+
bool RunCastFloat8ToFloat(long model_handle,const ENUM_FLOAT8_FORMAT fmt)
  {
   PrintFormat("TEST: %s(%s)",__FUNCTION__,EnumToString(fmt));
//---
   float test_data[15]   = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
   uchar data_float8[15] = {};
   if(!ArrayToFP8(data_float8,test_data,fmt))
     {
      Print("error in ArrayToFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   U<uchar> input_float8_values[3*5];
   U<float> output_float_values[3*5];
   float    test_data_float[];
//--- convert float8 to float
   if(!ArrayFromFP8(test_data_float,data_float8,fmt))
     {
      Print("error in ArrayFromFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   for(uint i=0; i<data_float8.Size(); i++)
     {
      input_float8_values[i].value=data_float8[i];
      PrintFormat("%d input value =%f  Hex float8 = %s  ushort value=%d",i,test_data_f
     }
   Print("ONNX input array: ",ArrayToString(input_float8_values));
//--- execute model (convert float8 to float using ONNX)
   if(!OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float8_values,output_float_values
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   Print("ONNX output array: ",ArrayToString(output_float_values));
//--- calculate error (compare ONNX and ArrayFromFP8 results)
   double sum_error=0.0;
   for(uint i=0; i<test_data.Size(); i++)
     {
      double delta=test_data_float[i]-(double)output_float_values[i].value;
      sum_error+=MathAbs(delta);
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("%d output float %f = %s difference=%f",i,output_float_values[i].val
     }
//---
   PrintFormat("%s(%s): sum_error=%f\n",__FUNCTION__,EnumToString(fmt),sum_error);
   return(true);
  }
```

Ver también

ArrayFromFP8, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

ArrayFromFP16

Permite copiar un array de tipo ushort en un array de tipo float o double con el formato establecido.

```
bool   ArrayFromFP16(
   const float&          dst_array[],        // a dónde copiamos
   const ushort&         src_array[],        // de dónde copiamos
   ENUM_FLOAT16_FORMAT   fmt                 // formato
   );
```

Sobrecarga para el tipo double

```
bool   ArrayFromFP16(
   const double&         dst_array[],        // a dónde copiamos
   const ushort&         src_array[],        // de dónde copiamos
   ENUM_FLOAT16_FORMAT   fmt                 // formato
   );
```

Parámetros

dst_array[]

[out] Array receptor del tipo float o double.

src_array[]

[in] Array fuente del tipo ushort.

fmt

[in] Formato de copiado de la enumeración ENUM_FLOAT16_FORMAT.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

Los formatos FLOAT16 y BFLOAT16 se definen en la enumeración ENUM_FLOAT16_FORMAT y se usan en MQL5 sola y exclusivamente para trabajar con modelos ONNX.

Si los parámetros de salida obtenidos al ejecutar la función OnnxRun son de tipo FLOAT16 y BFLOAT16, esta función permitirá convertir el resultado a arrays de tipo float o double.

FLOAT16, también conocido como formato de media precisión o "half-precision float", utiliza 16 bits para representar un número de coma flotante. Este formato ofrece un equilibrio entre precisión y eficiencia computacional. FLOAT16 se utiliza ampliamente en el aprendizaje profundo y las redes neuronales, donde se requiere un alto rendimiento al procesar grandes cantidades de datos. Este formato permite acelerar los cálculos reduciendo el tamaño de los números, lo cual resulta especialmente importante al entrenar redes neuronales profundas en unidades de procesamiento gráfico (GPU).

BFLOAT16 (o Brain Floating Point 16) también usa 16 bits, pero se diferencia de FLOAT16 en el modo de representación de los números. En este formato, se asignan 8 bits para representar el

© 2000-2025, MetaQuotes Ltd.

exponente y los 7 bits restantes se utilizan para representar la mantisa. Este formato se ha desarrollado para su uso en el aprendizaje profundo y la inteligencia artificial, especialmente en los procesadores Tensor Processing Unit (TPU) de Google. BFLOAT16 muestra un buen rendimiento en el entrenamiento de redes neuronales y puede utilizarse eficazmente para acelerar el cálculo.

Ejemplo: función del artículo Trabajamos con modelos ONNX en formatos float16 y float8

```
//+------------------------------------------------------------------+
//| RunCastFloat16ToDouble                                           |
//+------------------------------------------------------------------+
bool RunCastFloat16ToDouble(long model_handle)
  {
   PrintFormat("test=%s",__FUNCTION__);
   double test_data[12]= {1,2,3,4,5,6,7,8,9,10,11,12};
   ushort data_uint16[12];
   if(!ArrayToFP16(data_uint16,test_data,FLOAT_FP16))
     {
      Print("error in ArrayToFP16. error code=",GetLastError());
      return(false);
     }
   Print("test array:");
   ArrayPrint(test_data);
   Print("ArrayToFP16:");
   ArrayPrint(data_uint16);
   U<ushort> input_float16_values[3*4];
   U<double> output_double_values[3*4];
   float test_data_float[];
   if(!ArrayFromFP16(test_data_float,data_uint16,FLOAT_FP16))
     {
      Print("error in ArrayFromFP16. error code=",GetLastError());
      return(false);
     }
   for(int i=0; i<12; i++)
     {
      input_float16_values[i].value=data_uint16[i];
      PrintFormat("%d input value =%f  Hex float16 = %s  ushort value=%d",i,test_data_
     }
   Print("ONNX input array:");
   ArrayPrint(input_float16_values);
   bool res=OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float16_values,output_double
   if(!res)
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      return(false);
     }
   Print("ONNX output array:");
   ArrayPrint(output_double_values);
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   double sum_error=0.0;
   for(int i=0; i<12; i++)
     {
      double delta=test_data[i]-output_double_values[i].value;
      sum_error+=MathAbs(delta);
      PrintFormat("%d output double %f = %s  difference=%f",i,output_double_values[i].
     }
//---
   PrintFormat("test=%s   sum_error=%f",__FUNCTION__,sum_error);
//---
   return(true);
  }
```

Ver también

ArrayToFP16, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

ArrayFromFP8

Permite copiar un array de tipo uchar en un array de tipo float o double con el formato establecido.

```
bool   ArrayFromFP8(
   const float&          dst_array[],        // a dónde copiamos
   const uchar&          src_array[],        // de dónde copiamos
   ENUM_FLOAT8_FORMAT    fmt                 // formato
   );
```

Sobrecarga para el tipo double

```
bool   ArrayFromFP8(
   const double&         dst_array[],        // a dónde copiamos
   const uchar&          src_array[],        // de dónde copiamos
   ENUM_FLOAT8_FORMAT    fmt                 // formato
   );
```

Parámetros

dst_array[]

[out] Array receptor del tipo float o double.

src_array[]

[in] Array fuente del tipo uchar.

fmt

[in] Formato de copiado de la enumeración ENUM_FLOAT8_FORMAT.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

Todos los tipos de formato FP8 se definen en la enumeración ENUM_FLOAT8_FORMAT y se usan en MQL5 sola y exclusivamente para trabajar con modelos ONNX.

Si los parámetros de salida obtenidos al ejecutar la función OnnxRun son de tipo FP8 de la enumeración ENUM_FLOAT8_FORMAT, esta función permitirá convertir el resultado a arrays de tipo float o double.

El formato FP8 (coma flotante de 8 bits) es uno de los tipos de datos usados para representar números en coma flotante. En FP8, cada número se representa con 8 bits de datos, que suelen dividirse en tres componentes: signo, exponente y mantisa. Este formato ofrece un compromiso entre precisión y eficiencia de almacenamiento, lo cual lo hace atractivo para su uso en aplicaciones en las que es necesario ahorrar memoria y recursos computacionales.

Con su representación compacta de los números, el FP8 puede reducir los requisitos de memoria y acelerar los cálculos. Además, el FP8 puede ser útil para implementar operaciones de bajo nivel, como cálculos aritméticos y procesamiento de señales.

© 2000-2025, MetaQuotes Ltd.

Ejemplo: función del artículo Trabajamos con modelos ONNX en formatos float16 y float8

```
//+------------------------------------------------------------------+
//| RunCastFloat8Float                                               |
//+------------------------------------------------------------------+
bool RunCastFloat8ToFloat(long model_handle,const ENUM_FLOAT8_FORMAT fmt)
  {
   PrintFormat("TEST: %s(%s)",__FUNCTION__,EnumToString(fmt));
//---
   float test_data[15]   = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
   uchar data_float8[15] = {};
   if(!ArrayToFP8(data_float8,test_data,fmt))
     {
      Print("error in ArrayToFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   U<uchar> input_float8_values[3*5];
   U<float> output_float_values[3*5];
   float    test_data_float[];
//--- convert float8 to float
   if(!ArrayFromFP8(test_data_float,data_float8,fmt))
     {
      Print("error in ArrayFromFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   for(uint i=0; i<data_float8.Size(); i++)
     {
      input_float8_values[i].value=data_float8[i];
      PrintFormat("%d input value =%f  Hex float8 = %s  ushort value=%d",i,test_data_f
     }
   Print("ONNX input array: ",ArrayToString(input_float8_values));
//--- execute model (convert float8 to float using ONNX)
   if(!OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float8_values,output_float_values
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   Print("ONNX output array: ",ArrayToString(output_float_values));
//--- calculate error (compare ONNX and ArrayFromFP8 results)
   double sum_error=0.0;
   for(uint i=0; i<test_data.Size(); i++)
     {
      double delta=test_data_float[i]-(double)output_float_values[i].value;
      sum_error+=MathAbs(delta);
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("%d output float %f = %s difference=%f",i,output_float_values[i].val
     }
//---
   PrintFormat("%s(%s): sum_error=%f\n",__FUNCTION__,EnumToString(fmt),sum_error);
   return(true);
  }
```

Ver también

ArrayToFP8, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

Matrices y vectores

Una matriz es un array bidimensional de números double, float o complex.

Un vector es un array unidimensional de números double, float o complex. Al mismo tiempo, el vector no tiene signo sobre si es vertical u horizontal. Se determina partiendo del contexto de uso, por ejemplo, la operación vectorial Dot asume que el vector izquierdo es horizontal y el vector derecho es vertical. Si necesitamos una certeza absoluta, es posible usar matrices de una fila o de una columna, pero, en general, esto no resulta necesario.

Las matrices y vectores asignan la memoria para los datos dinámicamente. De hecho, las matrices y vectores son objetos que tienen ciertas propiedades, como el tipo de datos que contienen y las dimensiones. Las propiedades de las matrices y vectores se pueden obtener usando métodos como vector_a.Size(), matrix_b.Rows(), vector_c.Norm(), matrix_d.Cond(), etc. En este caso, podremos cambiar cualquier dimensionalidad.

Al crear e inicializar matrices, se usan los llamados métodos estáticos (son como métodos estáticos de una clase), por ejemplo: matrix::Eye(), matrix::Identity(), matrix::Ones(), vector::Ones(), matrix::Zeros(), vector::Zeros(), matrix::Full(), vector::Full(), matrix::Tri().

Por el momento, las operaciones con matrices y vectores no implican el uso del tipo de datos complejo; aún no se ha completado el trabajo en este plano.

MQL5 admite la transmisión de matrices y vectores a DLL. Esto permite que las funciones que usan este tipo de datos se importen desde bibliotecas externas.

Las matrices y vectores se transmiten a la DLL como el puntero a un búfer. Por ejemplo, para transmitir una matriz de tipo float, el parámetro correspondiente de la función DLL exportada deberá adoptar el puntero a un búfer de tipo float.

MQL5

```
#import "mmlib.dll"
bool sgemm(uint flags, matrix<float> &C, const matrix<float> &A, const matrix<float> &
#import
```

C++

```
extern "C" __declspec(dllexport) bool sgemm(UINT flags, float *C, const float *A, cons
```

Para procesar correctamente las matrices y vectores, además de sus búferes, es necesario transmitir sus tamaños.

Todos los métodos matriciales y vectoriales se enumeran a continuación en orden alfabético.

Función	Acción	Categoría
Activation	Calcula los valores de la función de activación y los escribe en el vector/matriz trasmitido	Aprendizaje automático
ArgMax	Retorna el índice del valor máximo	Estadísticas

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Categoría
ArgMin	Retorna el índice del valor mínimo	Estadísticas
ArgSort	Retorna el índice clasificado	Manipulación
Assign	Copia una matriz, vector o array con transformación automática	Inicialización
Average	Calcula la media ponderada de valores de matriz/vector	Estadísticas
Cholesky	Calcula la descomposición de Cholesky	Transformación
Clip	Limita los elementos de una matriz/vector a un rango dado de valores válidos	Manipulación
Col	Retorna un vector de columna. Escribe un vector en la columna especificada	Manipulación
Cols	Retorna el número de columnas en la matriz	Características
Compare	Compara los elementos de dos matrices/vectores con una precisión determinada	Manipulación
CompareByDigits	Compara si los elementos de dos matrices/vectores coinciden con una precisión de dígitos significativos	Manipulación
Cond	Calcula el número condicional de la matriz	Características
Convolve	Retorna la convolución lineal discreta de dos vectores	Productos
Copy	Retorna la copia de una matriz/vector dado	Manipulación
Concat	Concatena 2 submatrices en una sola matriz. Concatena 2 vectores en uno.	Manipulación
CopyIndicatorBuffer	Obtiene en un vector los datos del búfer indicado del indicador especificado en la cantidad especificada	Inicialización
CopyRates	Obtiene las series históricas de la estructura MqlRates en una	Inicialización

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Categoría
	matriz o vector del símbolo-periodo especificado en la cantidad especificada.	
CopyTicks	Obtiene en una matriz o vector los ticks de la estructura MqlTick	Inicialización
CopyTicksRange	Obtiene en una matriz o vector los ticks de la estructura MqlTick en el intervalo de fechas indicado	Inicialización
CorrCoef	Calcula el coeficiente de correlación de Pearson (coeficiente de correlación lineal)	Productos
Correlate	Calcula la correlación cruzada de dos vectores	Productos
Cov	Calcula la matriz de covarianza	Productos
CumProd	Retorna el producto acumulativo de los elementos de una matriz/vector incluidos los elementos a lo largo de un eje determinado	Estadísticas
CumSum	Retorna la suma acumulativa de los elementos de una matriz/vector incluidos los elementos a lo largo de un eje determinado	Estadísticas
Derivative	Calcula los valores de la derivada de la función de activación y los escribe en el vector/matriz transmitido	Aprendizaje automático
Det	Calcula el determinante de una matriz cuadrada no degenerada	Características
Diag	Extrae una diagonal o construye una matriz diagonal	Manipulación
Dot	Producto escalar de dos vectores	Productos
Eig	Calcula los valores propios y los vectores propios derechos de una matriz cuadrada	Transformación
EigVals	Calcula los valores propios de la matriz global	Transformación

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Categoría
Eye	Retorna una matriz con unidades en la diagonal y ceros en el resto de sitios	Inicialización
Fill	Rellena una matriz o vector existente con el valor dado	Inicialización
Flat	Permite referirse a un elemento de la matriz usando un único índice en lugar de dos	Manipulación
Full	Crea y retorna una nueva matriz rellenada con el valor dado	Inicialización
GeMM	Producto matricial total de dos matrices (General Matrix Multiply)	Productos
HasNan	Retorna el número de valores NaN en una matriz/vector	Manipulación
Hsplit	Partición horizontal de una matriz en varias submatrices. Lo mismo que Split con axis=0	Manipulación
Identity	Crea una matriz unitaria con el tamaño especificado	Inicialización
Init	Inicialización de una matriz o un vector	Inicialización
Inner	Producto interior de dos matrices	Productos
Inv	Calcula la inversa (multiplicativa) de una matriz cuadrada no degenerada usando el método de Jordan-Gauss.	Soluciones
Kron	Retorna el producto de Kronecker de dos matrices, una matriz y un vector, un vector y una matriz o dos vectores	Productos
LinearRegression	Calcula un vector/matriz con los valores de regresión lineal calculados	Estadísticas
Loss	Calcula los valores de la función de pérdida y los escribe en el vector/matriz trasmitido	Aprendizaje automático
LstSq	Retorna la solución de ecuaciones algebraicas lineales	Soluciones

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Categoría
	según el método de los mínimos cuadrados (para matrices no cuadradas o degeneradas)	
LU	Factorización LU de una matriz como producto de una matriz triangular inferior y una matriz triangular superior	Transformación
LUP	Factorización LUP con permutación parcial que se refiere a la descomposición LU solo con permutación de cadenas: PA=LU	Transformación
MatMul	Producto matricial de dos matrices	Productos
Max	Retrona el valor máximo de la matriz/vector	Estadísticas
Mean	Calcula la media aritmética de los valores de los elementos	Estadísticas
Median	Calcula la mediana de los elementos de la matriz/vector	Estadísticas
Min	Retorna el valor mínimo de la matriz/vector	Estadísticas
Norm	Retorna la norma de una matriz o vector	Características
Ones	Crea y retorna una nueva matriz rellenada con unidades	Inicialización
Outer	Calcula el producto exterior de dos matrices o dos vectores	Productos
Percentile	Retorna el percentil especificado de los valores o elementos de la matriz/vector a lo largo del eje especificado	Estadísticas
PInv	Calcula una matriz pseudoinversa según el método de Moore-Penrose	Soluciones
Power	Eleva una matriz cuadrada a una potencia entera	Productos
Prod	Retorna el producto de los elementos de una matriz/vector, que también	Estadísticas

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Categoría
	puede ejecutarse para un eje dado	
Ptp	Retorna el rango de valores de una matriz/vector o un eje de la matriz dado	Estadísticas
QR	Calcula la factorización QR de una matriz	Transformación
Quantile	Retorna el cuantil especificado de los valores de los elementos de la matriz/vector o los elementos a lo largo del eje especificado	Estadísticas
Random	Función estática. Crea y retorna una nueva matriz o vector relleno de valores aleatorios. Los valores aleatorios se generan uniformemente dentro de un rango determinado.	Inicialización
Rank	Retorna el rango de una matriz utilizando el método de Gauss	Características
RegressionMetric	Calcula la métrica de regresión como el error de desviación respecto a la línea de regresión trazada en el conjunto de datos especificado.	Estadísticas
Reshape	Cambia la forma de una matriz sin modificar sus datos	Manipulación
Resize	Retorna una nueva matriz con la forma y el tamaño modificados	Manipulación
Row	Devuelve un vector de filas. Escribe el vector en la fila especificada	Manipulación
Rows	Retorna el número de filas en la matriz	Características
Set	Establece el valor para el elemento del vector según el índice especificado	Manipulación
Size	Retorna el tamaño del vector	Características
SLogDet	Calcula el signo y el logaritmo del determinante de la matriz	Características

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Categoría
Solve	Resuelve una ecuación matricial lineal o un sistema de ecuaciones algebraicas lineales	Soluciones
Sort	Clasificación según el lugar	Manipulación
Spectrum	Calcula el espectro de una matriz como el conjunto de sus valores propios partiendo del producto AT*A	Características
Split	Partición de una matriz en varias submatrices	Manipulación
Std	Retorna la desviación estándar de los valores o elementos de la matriz/vector a lo largo de un eje determinado	Estadísticas
Sum	Retorna la suma de los elementos de una matriz/vector que también se puede ejecutar para un eje(s) dado(s)	Estadísticas
SVD	Descomposición singular de valores	Transformación
SwapCols	Intercambia las columnas de la matriz	Manipulación
SwapRows	Intercambia las filas de la matriz	Manipulación
Trace	Retorna la suma según las diagonales de la matriz	Características
Transpose	Transforma (intercambia los ejes) y retorna una matriz modificada	Manipulación
Tri	Construye una matriz con unidades en las diagonales especificada e inferior y ceros en el resto	Inicialización
TriL	Retorna una copia de la matriz con los elementos a cero sobre la k-ésima diagonal. Matriz triangular inferior	Manipulación
TriU	Retorna una copia de la matriz con los elementos a cero por debajo de la k-ésima diagonal. Matriz triangular superior	Manipulación

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Categoría
Var	Calcula la varianza de los valores de los elementos de la matriz/vector	Estadísticas
Vsplit	Partición vertical de una matriz en varias submatrices. Lo mismo que Split con axis=1	Manipulación
Zeros	Crea y retorna una nueva matriz rellenada con ceros	Inicialización

© 2000-2025, MetaQuotes Ltd.

Tipos de matrices y vectores

matrix y vector son tipos de datos especiales en MQL5, diseñados para realizar operaciones de álgebra lineal. Existen los siguientes tipos de datos:

- matrix — matriz que contiene elementos de tipo double
- matrixf — matriz que contiene elementos del tipo float
- matrix — matriz que contiene elementos del tipo complex
- vector — vector que contiene elementos del tipo double
- vectorf — vector que contiene elementos del tipo float
- vectorc — vector que contiene elementos del tipo complex

Para su uso en funciones de plantilla, podemos utilizar la entrada matrix<double>, matrix<float>, matrix<complex>, vector<double>, vector<float>, vector<complex> en lugar de los tipos correspondientes.

MQL5 admite la transmisión de matrices y vectores a DLL. Esto permite que las funciones que usan este tipo de datos se importen desde bibliotecas externas.

Las matrices y vectores se transmiten a la DLL como el puntero a un búfer. Por ejemplo, para transmitir una matriz de tipo float, el parámetro correspondiente de la función DLL exportada deberá adoptar el puntero a un búfer de tipo float.

MQL5

```
#import "mmlib.dll"
bool sgemm(uint flags, matrix<float> &C, const matrix<float> &A, const matrix<float> &
#import
```

C++

```
extern "C" __declspec(dllexport) bool sgemm(UINT flags, float *C, const float *A, cons
```

Para procesar correctamente las matrices y vectores, además de sus búferes, es necesario transmitir sus tamaños.

Métodos de inicialización de matrices y vectores

Función	Acción
Assign	Copia una matriz, vector o array con transformación automática
CopyRates	Obtiene las series históricas de la estructura MqlRates en una matriz o vector del símbolo-periodo especificado en la cantidad especificada.
Eye	Retorna una matriz con unidades a lo largo de la diagonal y ceros en el resto
Identity	Crea una matriz unitaria con el tamaño especificado
Ones	Crea y retorna una nueva matriz rellenada con unidades
Zeros	Crea y retorna una nueva matriz rellenada con ceros

© 2000-2025, MetaQuotes Ltd.

Función	Acción
Full	Crea y retorna una nueva matriz rellenada con los valores dados
Tri	Crea una matriz con unidades en la diagonal indicada y por debajo de la misma, y ceros en el resto.
Init	Inicializa una matriz o vector
Fill	Rellena una matriz o vector existente con el valor dado

© 2000-2025, MetaQuotes Ltd.

Enumeraciones para trabajar con matrices y vectores

Esta sección describe las enumeraciones utilizadas en varios métodos de trabajo con matrices y vectores.

ENUM_AVERAGE_MODE

Enumeración de los tipos de promediación.

Identificador	Descripción
AVERAGE_NONE	Ninguna promediación. Los resultados se ofrecen por separado para cada etiqueta
AVERAGE_BINARY	Resultado de la etiqueta 1 según la clasificación binaria
AVERAGE_MICRO	Resultado según la matriz de errores promediada (confusion matrix)
AVERAGE_MACRO	Resultado promediado de los resultados de las matrices de error de cada etiqueta
AVERAGE_WEIGHTED	Resultado promedio ponderado

ENUM_VECTOR_NORM

Enumeración de normas vectoriales para vector::Norm.

Identificador	Descripción
VECTOR_NORM_INF	Norma infinita
VECTOR_NORM_MINUS_INF	Norma infinita negativa
VECTOR_NORM_P	Norma P

ENUM_MATRIX_NORM

Enumeración de normas matriciales para matrix::Norm y para obtener el número de condiciones de la matriz matrix::Cond.

Identificador	Descripción
MATRIX_NORM_FROBENIUS	Norma de Frobenius
MATRIX_NORM_SPECTRAL	Norma espectral
MATRIX_NORM_NUCLEAR	Norma nuclear

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
MATRIX_NORM_INF	Norma infinita
MATRIX_NORM_P1	Norma P1
MATRIX_NORM_P2	Norma P2
MATRIX_NORM_MINUS_INF	Norma infinita negativa
MATRIX_NORM_MINUS_P1	Norma negativa P1
MATRIX_NORM_MINUS_P2	Norma negativa P2

ENUM_VECTOR_CONVOLVE

Enumeración para la convolución vector::Convolve la correlación cruzada vector::Correlate.

Identificador	Descripción
VECTOR_CONVOLVE_FULL	Convolución completa
VECTOR_CONVOLVE_SAME	Convolución con el tipo same
VECTOR_CONVOLVE_VALID	Convolución con el tipo valid

ENUM_REGRESSION_METRIC

Enumeración de métricas de regresión vector::RegressionMetric.

Identificador	Descripción
REGRESSION_MAE	Error absoluto medio
REGRESSION_MSE	Error cuadrático medio
REGRESSION_RMSE	Raíz del error cuadrático medio
REGRESSION_R2	Cuadrado R
REGRESSION_MAPE	Error porcentual absoluto medio
REGRESSION_MSPE	Error porcentual cuadrático medio
REGRESSION_RMSLE	Raíz del error logarítmico cuadrático medio
REGRESSION_SMAPE	Error porcentual absoluto medio simétrico
REGRESSION_MAXE	Error absoluto máximo
REGRESSION_MEDE	Error absoluto mediano
REGRESSION_MPD	Desviación media de Poisson

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
REGRESSION_MGD	Desviación gamma media
REGRESSION_EXPV	Varianza explicada

ENUM_CLASSIFICATION_METRIC

Enumeración de métricas para tareas de clasificación.

Identificador	Descripción
CLASSIFICATION_ACCURACY	Calidad del modelo en cuanto a la fidelidad de las predicciones en todas las clases
CLASSIFICATION_AVERAGE_PRECISION	Precisión promediada del modelo
CLASSIFICATION_BALANCED_ACCURACY	Precisión equilibrada de las precciones
CLASSIFICATION_F1	Medida F1. Media armónica entre la precisión (precision) y la completitud (recall) del modelo
CLASSIFICATION_JACCARD	Medida de Jaccard
CLASSIFICATION_PRECISION	Precisión del modelo en la predicción de resultados verdaderamente positivos para la clase objetivo
CLASSIFICATION_RECALL	Completitud del modelo
CLASSIFICATION_ROC_AUC	Área bajo la curva de error
CLASSIFICATION_TOP_K_ACCURACY	Frecuencia de aparición de la etiqueta correcta en el top de k etiquetas predichas

ENUM_LOSS_FUNCTION

Enumeración para calcular la función de pérdida vector::Loss.

Identificador	Descripción
LOSS_MSE	Error medio cuadrático
LOSS_MAE	Error absoluto medio
LOSS_CCE	Entropía cruzada categórica
LOSS_BCE	Entropía cruzada binaria
LOSS_MAPE	Error porcentual absoluto medio
LOSS_MSLE	Error logarítmico cuadrático medio
LOSS_KLD	Divergencia de Kullback-Leibler

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
LOSS_COSINE	Similitud/proximidad del coseno
LOSS_POISSON	Función de pérdida de Poisson
LOSS_HINGE	Función de pérdida lineal a trozos (Hinge loss)
LOSS_SQ_HINGE	Función de pérdida lineal cuadrada a trozos
LOSS_CAT_HINGE	Función de pérdida lineal categórica a trozos
LOSS_LOG_COSH	Logaritmo del coseno hiperbólico
LOSS_HUBER	Función de pérdida de Hubert

ENUM_ACTIVATION_FUNCTION

Enumeración para la función de activación vector::Activation y la derivada de la función de activación vector::Derivative.

Identificador	Descripción	Parámetros
AF_NONE	La función de activación no se utiliza, el valor de entrada se transmite a la salida	
AF_ELU	Unidad lineal exponencial	
AF_EXP	Exponencial	
AF_GELU	Unidad lineal de error Gauss	
AF_HARD_SIGMOID	Sigmoide duro	
AF_LINEAR	Lineal	
AF_LRELU	Rectificador lineal con "fuga" (Leaky ReLU)	
AF_RELU	Transformación lineal truncada ReLU	
AF_SELU	Función lineal exponencial escalada (Scaled ELU)	
AF_SIGMOID	Sigmoide	
AF_SOFTMAX	Softmax	

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Parámetros
AF_SOFTPLUS	Softplus	
AF_SOFTSIGN	Softsign	
AF_SWISH	Función Swish	
AF_TANH	Tangente hiperbólica	
AF_TRELU	Rectificador lineal con umbral	

ENUM_SORT_MODE

Enumeración de los tipos de clasificación de la función Sort.

Identificador	Descripción
SORT_ASCENDING	Clasificación por orden ascendente
SORT_DESCENDING	Clasificación por orden descendente

ENUM_MATRIX_AXIS

Enumeración para indicar el eje en todas las funciones estadísticas para las matrices.

Identificador	Descripción
AXIS_NONE	El eje no está definido, el cálculo se realiza sobre todos los elementos de la matriz, como si se tratara de un vector (véase el método Flat).
AXIS_HORZ	Eje horizontal
AXIS_VERT	Eje vertical

© 2000-2025, MetaQuotes Ltd.

Inicialización

Existen varias formas de declarar e inicializar matrices y vectores.

Función	Acción
Assign	Copia una matriz, vector o array con transformación automática
CopyIndicatorBuffer	Obtiene en un vector los datos del búfer indicado del indicador especificado en la cantidad especificada
CopyRates	Obtiene las series históricas de la estructura MqlRates en una matriz o vector del símbolo-periodo especificado en la cantidad especificada.
CopyTicks	Obtiene en una matriz o vector los ticks de la estructura MqlTick
CopyTicksRange	Obtiene en una matriz o vector los ticks de la estructura MqlTick en el intervalo de fechas indicado
Eye	Retorna una matriz con unidades a lo largo de la diagonal y ceros en el resto
Identity	Crea una matriz unitaria con el tamaño especificado
Ones	Crea y retorna una nueva matriz rellenada con unidades
Zeros	Crea y retorna una nueva matriz rellenada con ceros
Full	Crea y retorna una nueva matriz rellenada con los valores dados
Tri	Crea una matriz con unidades en la diagonal indicada y por debajo de la misma, y ceros en el resto.
Init	Inicializa una matriz o vector
Fill	Rellena una matriz o vector existente con el valor dado
Random	Función estática. Crea y retorna una nueva matriz o vector relleno de valores aleatorios. Los valores aleatorios se generan uniformemente dentro de un rango determinado.

Declaración de tamaño no especificado (sin asignación de memoria para los datos):

```
matrix          matrix_a;   // matriz del tipo double
matrix<double>  matrix_a1;  // otra forma de declarar una matriz double, resulta adec
matrixf         matrix_a2;  // matriz de tipo float
matrix<float>   matrix_a3;  // matriz de tipo float
vector          vector_a;   // vecto de tipo double
vector<double>  vector_a1;
vectorf         vector_a2;  // vecto de tipo float
vector<float>   vector_a3;
```

© 2000-2025, MetaQuotes Ltd.

Declaración con declaración del tamaño (con asignación de memoria para datos, pero sin ninguna inicialización):

```
  matrix         matrix_a(128,128);           // como constantes se pueden especificar
  matrix<double> matrix_a1(InpRows,InpCols);  // como variables
  matrixf        matrix_a2(1,128);            // análogo del vector horizontal
  matrix<float>  matrix_a3(InpRows,1);        // análogo del vector vertical
  vector         vector_a(256);
  vector<double> vector_a1(InpSize);
  vectorf        vector_a2(SomeFunc());       // la función SomeFunc retorna un número
  vector<float>  vector_a3(InpSize+16);       // como parámetro se puede utilizar una
```

Declaración con inicialización (los tamaños de las matrices y vectores en este caso estarán determinados por la secuencia de inicialización):

```
  matrix         matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
  matrix<double> matrix_a1=matrix_a;                    // las matrices deberán ser
  matrixf        matrix_a2={{1,0,0},{0,1,0},{0,0,1}};
  matrix<float>  matrix_a3={{1,2},{3,4}};
  vector         vector_a={-5,-4,-3,-2,-1,0,1,2,3,4,5};
  vector<double> vector_a1={1,5,2.4,3.3};
  vectorf        vector_a2={0,1,2,3};
  vector<float>  vector_a3=vector_a2;                   // los vectores deberán ser
```

Declaración con inicialización:

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
...

  matrix  matrix_a(size_m,size_k,MatrixArange,-M_PI,0.1); // primero, se crea una matr
  matrixf matrix_a1(10,20,MatrixArange);                  // después de crear la matri
  vector  vector_a(size,VectorArange,-10.0);              // una vez creado el vector,
  vectorf vector_a1(128,VectorArange);
```

© 2000-2025, MetaQuotes Ltd.

Debemos tener en cuenta que las dimensiones de la matriz o vector se pueden cambiar, ya que la memoria de datos siempre es dinámica.

Métodos estáticos

Métodos estáticos para crear matrices y vectores del tamaño especificado, inicializados de cierta manera:

```
matrix          matrix_a =matrix::Eye(4,5,1);
matrix<double> matrix_a1=matrix::Full(3,4,M_PI);
matrixf         matrix_a2=matrixf::Identity(5,5);
matrixf<float> matrix_a3=matrixf::Ones(5,5);
matrix          matrix_a4=matrix::Tri(4,5,-1);
vector          vector_a =vector::Ones(256);
vectorf         vector_a1=vector<float>::Zeros(16);
vector<float>   vector_a2=vectorf::Full(128,float_value);
```

Métodos para inicializar las matrices y vectores ya creados:

```
matrix  matrix_a;
matrix_a.Init(size_m,size_k,MatrixArange,-M_PI,0.1);
matrixf matrix_a1(3,4);
matrix_a1.Init(10,20,MatrixArange);
vector  vector_a;
vector_a.Init(128,VectorArange);
vectorf vector_a1(10);
vector_a1.Init(vector_size,VectorArange,start_value,step);

matrix_a.Fill(double_value);
vector_a1.Fill(FLT_MIN);
matrix_a1.Identity();
```

© 2000-2025, MetaQuotes Ltd.

Assign

Copia una matriz, vector o array con transformación automática.

```
bool matrix::Assign(
  const matrix<T>  &mat     // matriz a copiar
   );
bool matrix::Assign(
  const void       &array[] // matriz a copiar
   );
bool vector::Assign(
  const vector<T>  &vec     // vector a copiar
   );
bool vector::Assign(
  const void       &array[] // matriz a copiar
   );
```

Parámetros

m, v o array

[in] Matriz, vector o array del que se copian los valores.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

A diferencia de Copy, el método Assign también permite copiar arrays. Además, se realiza automáticamente una conversión de tipo, y la matriz o vector resultante se ajustará al tamaño del array copiado.

Ejemplo:

```
//--- copiado de matrices
  matrix a= {{2, 2}, {3, 3}, {4, 4}};
  matrix b=a+2;
  matrix c;
  Print("matrix a \n", a);
  Print("matrix b \n", b);
  c.Assign(b);
  Print("matrix c \n", a);

//--- copiando el array en una matriz
  matrix double_matrix=matrix::Full(2,10,3.14);
  Print("double_matrix before Assign() \n", double_matrix);
  int int_arr[5][5]= {{1, 2}, {3, 4}, {5, 6}};
  Print("int_arr: ");
  ArrayPrint(int_arr);
```

© 2000-2025, MetaQuotes Ltd.

```
double_matrix.Assign(int_arr);
Print("double_matrix after Assign(int_arr) \n", double_matrix);
/*
 matrix a
 [[2,2]
  [3,3]
  [4,4]]
 matrix b
 [[4,4]
  [5,5]
  [6,6]]
 matrix c
 [[2,2]
  [3,3]
  [4,4]]

 double_matrix before Assign()
 [[3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]
  [3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]]

 int_arr:
     [,0][,1][,2][,3][,4]
 [0,]   1   2   0   0   0
 [1,]   3   4   0   0   0
 [2,]   5   6   0   0   0
 [3,]   0   0   0   0   0
 [4,]   0   0   0   0   0

 double_matrix after Assign(int_arr)
 [[1,2,0,0,0]
  [3,4,0,0,0]
  [5,6,0,0,0]
  [0,0,0,0,0]
  [0,0,0,0,0]]

*/
```

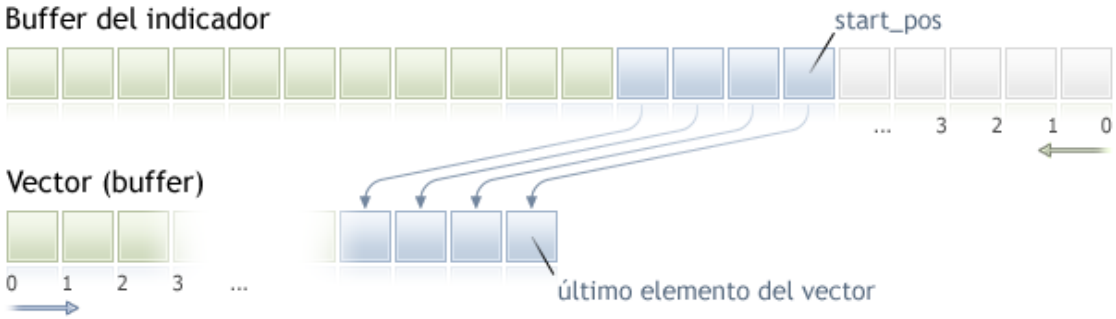
Ver también

Copy

© 2000-2025, MetaQuotes Ltd.

CopyIndicatorBuffer

Obtiene en un vector los datos del búfer indicado del indicador especificado en la cantidad especificada.



Los elementos de los datos a copiar (el búfer de indicador con el índice buffer_index) se cuentan desde la posición de inicio partiendo del presente hacia el pasado, es decir, una posición de inicio igual a 0 indicará la barra actual (valor del indicador para la barra actual).

Al copiar una cantidad desconocida de datos, deberemos declarar el vector sin especificar el tamaño (sin asignar memoria a los datos), porque la función CopyBuffer() intentará asignar el tamaño del vector receptor al tamaño de los datos copiados.

Si queremos copiar los valores de los indicadores parcialmente, deberemos utilizar para ello un vector intermedio en el que se copie la cantidad necesaria, y ya desde este vector intermedio realizar el copiado elemento a elemento del número necesario de valores en los lugares apropiados del vector receptor.

Si se va a copiar una cantidad de datos conocida de antemano, será mejor declarar previamente el tamaño del vector para evitar sobreasignaciones de memoria innecesarias.

Los datos del vector se copiarán de forma que el elemento más antiguo se encuentre al principio de la memoria física asignada para el vector. Existen 3 variantes de la función.

Referenciación según la posición inicial y el número de elementos necesarios

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle del indicador
   ulong     buffer_index,         // número de búfer del indicador
   ulong     start_pos,            // desde dónde comenzamos
   ulong     count                 // cuánto copiamos
   );
```

Referenciación según la fecha inicial y el número de elementos necesarios

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle del indicador
   ulong     buffer_index,         // número de búfer del indicador
   datetime  start_time,           // desde qué fecha
   ulong     count                 // cuánto copiamos
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Referenciación según la fecha inicial y final del intervalo temporal requerido.

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle del indicador
   ulong     buffer_index,         // número de búfer del indicador
   datetime  start_time,           // desde qué fecha
   datetime  stop_time             // hasta qué fecha
   );
```

Parámetros

indicator_handle

[in] Manejador del indicador obtenido con la función de indicador correspondiente.

buffer_index

[in] Número del búfer de indicador.

start_pos

[in] Número del primer elemento copiado.

count

[in] Número de elementos copiados.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

Valor retornado

Retorna true en caso de éxito, o false en caso de error.

Observación

Al solicitar los datos desde el indicador, si las series temporales solicitadas aún no han sido construidas o deben ser cargadas desde el servidor, la función retornará de inmediato false, pero, en este caso, el propio proceso de carga/construcción será iniciado.

Al solicitar los datos desde un experto o script, se inicializará la carga desde el servidor, si el terminal no dispone de estos datos a nivel local; o bien comenzará la construcción de la serie temporal necesaria, si los datos se pueden construir a partir de la historia local, pero aún no están preparados. La función retornará la cantidad de datos que estarán listos cuando expire el tiempo de espera.

Ver también

CopyBuffer

© 2000-2025, MetaQuotes Ltd.

CopyRates

Obtiene las series históricas de la estructura MqlRates en una matriz o vector del símbolo-periodo especificado en la cantidad especificada. Los elementos se contarán partiendo de la posición inicial desde el presente hacia el pasado, es decir, si la posición inicial es igual a 0, esto indicará la barra actual.

En este caso, además, los datos se copiarán de forma que el elemento más antiguo se sitúe al principio de la matriz/vector. Existen 3 variantes del método.

Referenciación según la posición inicial y el número de elementos necesarios

```
bool  matrix::CopyRates(
   string           symbol,       // nombre del símbolo
   ENUM_TIMEFRAMES  period,       // periodo
   ulong            rates_mask,   // combinación de banderas para indicar las series s
   ulong            start,        // número de la barra inicial desde la que comenzare
   ulong            count         // cuántas copiaremos
   );
```

Referenciación según la fecha inicial y el número de elementos necesarios

```
bool  matrix::CopyRates(
   string           symbol,       // nombre del símbolo
   ENUM_TIMEFRAMES  period,       // periodo
   ulong            rates_mask,   // combinación de banderas para indicar las series s
   datetime         from,         // desde qué fecha
   ulong            count         // cuántas copiaremos
   );
```

Referenciación según la fecha inicial y final del intervalo temporal requerido.

```
bool  matrix::CopyRates(
   string           symbol,       // nombre del símbolo
   ENUM_TIMEFRAMES  period,       // periodo
   ulong            rates_mask,   // combinación de banderas para indicar las series s
   datetime         from,         // desde qué fecha
   datetime         to            // hasta qué fecha
   );
```

Para los vectores, los métodos son iguales.

Referenciación según la posición inicial y el número de elementos necesarios

```
bool  vector::CopyRates(
   string           symbol,       // nombre del símbolo
   ENUM_TIMEFRAMES  period,       // periodo
   ulong            rates_mask,   // combinación de banderas para indicar las series s
   ulong            start,        // número de la barra inicial desde la que comenzare
   ulong            count         // cuántas copiaremos
   );
```

© 2000-2025, MetaQuotes Ltd.

Referenciación según la fecha inicial y el número de elementos necesarios

```
bool  vector::CopyRates(
   string           symbol,       // nombre del símbolo
   ENUM_TIMEFRAMES  period,       // periodo
   ulong            rates_mask,   // combinación de banderas para indicar las series s
   datetime         from,         // desde qué fecha
   ulong            count         // cuántas copiaremos
   );
```

Referenciación según la fecha inicial y final del intervalo temporal requerido.

```
bool  vector::CopyRates(
   string           symbol,       // nombre del símbolo
   ENUM_TIMEFRAMES  period,       // periodo
   ulong            rates_mask,   // combinación de banderas para indicar las series s
   datetime         from,         // desde qué fecha
   datetime         to            // hasta qué fecha
   );
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo.

rates_mask

[in] Combinación de banderas de la enumeración ENUM_COPY_RATES que indican el tipo de series solicitadas. Al copiar a un vector, podemos indicar solo un valor de la enumeración ENUM_COPY_RATES, de lo contrario, ocurrirá un error.

start

[in] Número del primer elemento copiado.

count

[in] Número de elementos copiados.

from

[in] Hora de la barra correspondiente al primer elemento.

to

[in] Hora de la barra correspondiente al último elemento.

Valor retornado

Retorna true en caso de éxito, o false en caso de error.

Observación

© 2000-2025, MetaQuotes Ltd.

Si el intervalo de los datos solicitados se encuentra totalmente fuera de los datos en el servidor, la función retornará false. Si los datos solicitados se encuentran fuera de TERMINAL_MAXBARS (número máximo de barras en el gráfico), la función también retornará false.

Al solicitar los datos desde un experto o script, se inicializará la carga desde el servidor, si el terminal no dispone de estos datos a nivel local; o bien comenzará la construcción de la serie temporal necesaria, si los datos se pueden construir a partir de la historia local, pero aún no están preparados. La función retornará la cantidad de datos que estén preparados al momento de finalización del timeout, pero la carga de la historia continuará, y con la siguiente solicitud análoga, la función retornará ya más datos.

Al solicitar los datos según la fecha de inicio y el número de elementos necesarios, solo se retornarán los datos con una fecha inferior (anterior) o igual a la fecha especificada. El intervalo se establece y se considera con una precisión que abarca hasta el segundo más próximo. Es decir, la fecha de apertura de cualquier barra para la que se retorna un valor (volumen, spread, precio Open, High, Low, Close u hora de apertura), será siempre igual o inferior a la especificada.

Al solicitar los datos en el intervalo de fechas especificado, solo se retornarán los datos comprendidos en el intervalo solicitado, y el intervalo se especificará y considerará hasta el segundo más próximo. Esto significará que la hora de apertura de cualquier barra para la que se retorna un valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close u hora de apertura), estará siempre dentro del intervalo solicitado.

Así, si el día de la semana actual es el sábado, al intentar copiar los datos en el marco temporal semanal con *start_time=Last_Tuesday* y *stop_time=Last_Friday* la función retornará 0, porque la hora de apertura del marco temporal semanal caerá siempre en domingo, pero ninguna barra semanal estará dentro del intervalo especificado.

Si deseamos obtener el valor correspondiente a la barra actual no finalizada, podremos utilizar la primera forma de llamada, especificando *start_pos*=0 y *count*=1..

ENUM_COPY_RATES

La enumeración ENUM_COPY_RATES contiene las banderas para especificar el tipo de datos que se transmitirán a la matriz o array. La combinación de banderas permite recuperar varias series de la historia en una sola solicitud. El orden de las filas en la matriz se corresponderá con el orden de los valores en la enumeración ENUM_COPY_RATES, es decir, la fila con los datos High siempre estará por encima de la fila con los datos Low en la matriz.

Identificador	Valor	Descripción
COPY_RATES_OPEN	1	Serie de precios Open
COPY_RATES_HIGH	2	Serie de precios High
COPY_RATES_LOW	4	Serie de precios Low
COPY_RATES_CLOSE	8	Serie de precios Close
COPY_RATES_TIME	16	Serie Time (hora de apertura de la barra) La obtención de la hora en float de un vector o matriz (vectord y matrixf)

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor	Descripción
		provocará una pérdida de ~100 segundos, dado que la precisión de float está fuertemente limitada, y los números enteros superiores a 1<<24 no pueden representarse con exactitud en float.
COPY_RATES_VOLUME_TICK	32	Volúmenes de ticks
COPY_RATES_VOLUME_REAL	64	Volúmenes comerciales
COPY_RATES_SPREAD	128	Spread
Combinación		
COPY_RATES_OHLC	15	Series Open, High, Low y Close
COPY_RATES_OHLCT	31	Series Open, High, Low, Close y Time
Ubicación de los datos		
COPY_RATES_VERTICAL	32768	Las series se copian en la matriz a lo largo del eje vertical.Esto significa que los valores de la serie obtenidos en la matriz se ordenan verticalmente, es decir, los datos más antiguos están en la primera fila y los más recientes en la última fila de la matriz. Por defecto, al copiar en la matriz, las series se añaden a lo largo del eje horizontal. La bandera solo es relevante cuando se copia en una matriz.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- obtenemos las cotizaciones en la matriz
  matrix matrix_rates;
  if(matrix_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLCT, 1, 10))
    Print("matrix rates: \n", matrix_rates);
  else
    Print("matrix_rates.CopyRates failed. Error ", GetLastError());
//--- comprobación
  MqlRates mql_rates[];
  if(CopyRates(Symbol(), PERIOD_CURRENT, 1, 10, mql_rates)>0)
   {
    Print("mql_rates array:");
```

© 2000-2025, MetaQuotes Ltd.

```
    ArrayPrint(mql_rates);
   }
  else
    Print("CopyRates(Symbol(), PERIOD_CURRENT,1, 10, mql_rates). Error ", GetLastError
//--- obtenemos las cotizaciones en el vector = llamada errónea
  vector vector_rates;
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLC, 1, 15))
    Print("vector_rates COPY_RATES_OHLC: \n", vector_rates);
  else
    Print("vector_rates.CopyRates COPY_RATES_OHLC failed. Error ", GetLastError());
//--- obtenemos los precios de cierre en un vector
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_CLOSE, 1, 15))
    Print("vector_rates COPY_RATES_CLOSE: \n", vector_rates);
  else
    Print("vector_rates.CopyRates failed. Error ", GetLastError());
 };
/*
   matrix rates:
   [[0.99686,0.99638,0.99588,0.99441,0.99464,0.99594,0.99698,0.99758,0.99581,0.995280
    [0.99708,0.99643,0.99591,0.9955000000000001,0.99652,0.99795,0.99865,0.99764,0.996
    [0.9961100000000001,0.99491,0.99426,0.99441,0.99448,0.99494,0.9964499999999999,0.9
    [0.99641,0.99588,0.99441,0.99464,0.99594,0.99697,0.99758,0.99581,0.995280000000000
    [1662436800,1662440400,1662444000,1662447600,1662451200,1662454800,1662458400,1662
   mql_rates array:
                    [time]   [open]   [high]    [low] [close] [tick_volume] [spread] [rea
   [0] 2022.09.06 04:00:00 0.99686 0.99708 0.99611 0.99641           4463        0
   [1] 2022.09.06 05:00:00 0.99638 0.99643 0.99491 0.99588           4519        0
   [2] 2022.09.06 06:00:00 0.99588 0.99591 0.99426 0.99441           3060        0
   [3] 2022.09.06 07:00:00 0.99441 0.99550 0.99441 0.99464           3867        0
   [4] 2022.09.06 08:00:00 0.99464 0.99652 0.99448 0.99594           5280        0
   [5] 2022.09.06 09:00:00 0.99594 0.99795 0.99494 0.99697           7227        0
   [6] 2022.09.06 10:00:00 0.99698 0.99865 0.99645 0.99758          10130        0
   [7] 2022.09.06 11:00:00 0.99758 0.99764 0.99472 0.99581           7012        0
   [8] 2022.09.06 12:00:00 0.99581 0.99604 0.99360 0.99528           6166        0
   [9] 2022.09.06 13:00:00 0.99528 0.99570 0.99220 0.99259           6950        0
   vector_rates.CopyRates COPY_RATES_OHLC failed. Error 4003
   vector_rates COPY_RATES_CLOSE:
   [0.9931,0.99293,0.99417,0.99504,0.9968399999999999,0.99641,0.99588,0.99441,0.99464,
*/
```

Ver también

Acceso a las series temporales e indicadores, CopyRates

© 2000-2025, MetaQuotes Ltd.

CopyTicks

Obtiene los ticks de la estructura MqlTick en una matriz o vector.El conteo de elementos desde la posición de inicio se realiza del pasado hacia el presente, es decir, el tick con el índice 0 será el más antiguo. Para analizar el tick deberemos comprobar el campo *flags*, que indica exactamente qué ha cambiado en el tick en cuestión.

```
bool  matrix::CopyTicks(
   string           symbol,              // nombre del símbolo
   ulong            flags,               // bandera que determina el tipo de ticks c
   ulong            from_msc,            // hora desde la que se solicitan los ticks
   ulong            count                // número de ticks que debemos obtener
   );
```

Método del vector

```
bool  vector::CopyTicks(
   string           symbol,              // nombre del símbolo
   ulong            flags,               // bandera que determina el tipo de ticks c
   ulong            from_msc,            // hora desde la que se solicitan los ticks
   ulong            count                // número de ticks que debemos obtener
   );
```

Parámetros

symbol

[in] Símbolo.

flags

[in] Combinación de banderas de la enumeración ENUM_COPY_TICKS que indican la composición de los datos solicitados. Al copiar a un vector, podemos indicar solo un valor de la enumeración ENUM_COPY_TICKS, de lo contrario, ocurrirá un error.

from_msc

[in] Hora a partir de la cual se solicitan los ticks. Se especifica en milisegundos a partir del 01.01.1970. Si el parámetro es from_msc=0, se ofrecerán los últimos count de los ticks.

count

[in] Número de ticks solicitados. Si no se especifican los parámetros from_msc y count, se registrarán todos los últimos ticks disponibles, pero no más de 2000.

Valor retornado

Retorna true en caso de éxito, o false en caso de error.

Observación

La primera llamada a CopyTicks() inicia la sincronización de la base de datos de ticks almacenada en el disco duro para un símbolo determinado. Si no hay suficientes ticks en la base de datos local, los ticks que falten se cargarán automáticamente desde el servidor comercial. En este caso, los ticks del momento ***de_msc*** especificado en CopyTicks() se sincronizarán con el momento actual. Después

© 2000-2025, MetaQuotes Ltd.

de eso, todos los ticks entrantes en el símbolo dado irán a la base de ticks y la mantendrán sincronizada hasta la fecha.

Si no se especifican los parámetros *from_msc* y *count*, se escribirán en la matriz/vector todos los ticks disponibles, pero no más de 2000.

En los indicadores, el método CopyTicks() retornará el resultado de inmediato: al realizar la llamada desde un indicador, CopyTick() devolverá directamente los ticks disponibles por símbolo, y también activará la sincronización de la base de datos de ticks si no hay datos suficientes. Todos los indicadores de un símbolo trabajan en un hilo común, por lo que el indicador no tendrá derecho a esperar a que finalice la sincronización. Una vez completada la sincronización, la siguiente llamada a CopyTicks() retornará todos los ticks solicitados. La función OnCalculate() de los indicadores se llamará después de cada tick obtenido.

En los expertos y scripts, el método CopyTicks() puede esperar el resultado hasta 45 segundos: a diferencia de los indicadores cada experto y script trabajará en su propio hilo, y por lo tanto podrá esperar la sincronización hasta 45 segundos. Si en este tiempo no se sincronizan los ticks en la cantidad necesaria, CopyTicks() retornará solo los ticks disponibles por timeout, y la sincronización continuará. La función OnTick() en el asesor no es un manejador de ticks, solo notifica al asesor sobre los cambios en el mercado. Los cambios pueden darse por lotes: varios ticks pueden llegar al terminal al mismo tiempo, pero la función OnTick() será llamada una sola vez, para notificar al asesor experto la última situación del mercado.

Velocidad de ejecución: el terminal almacena para cada símbolo los 4096 últimos ticks en la caché para el acceso rápido (para los símbolos con la pila en ejecución, 65536 ticks), las peticiones a estos datos se ejecutarán más rápidamente. Al solicitar los ticks de la sesión comercial actual fuera de la caché, CopyTicks() accederá a los ticks ya almacenados en la memoria del terminal, estas solicitudes tardarán más tiempo en ejecutarse. Las peticiones de los ticks de otros días serán las más lentas, ya que en este caso, los datos se leerán desde el disco.

ENUM_COPY_TICKS

La enumeración ENUM_COPY_TICKS contiene las banderas para especificar el tipo de datos que se transmitirán a la matriz o array. La combinación de banderas permite recuperar varias series de la historia en una sola solicitud. El orden de las filas en la matriz se corresponderá con el orden de los valores en la enumeración ENUM_COPY_TICKS, es decir, la fila con los datos High siempre estará por encima de la fila con los datos Low en la matriz.

Identificador	Valor	Descripción
COPY_TICKS_INFO	1	
COPY_TICKS_TRADE	2	
COPY_TICKS_ALL	3	
COPY_TICKS_TIME_MS	1<<8	Hora del tick en milisegundos
COPY_TICKS_BID	1<<9	Precio Bid
COPY_TICKS_ASK	1<<10	Precio Ask
COPY_TICKS_LAST	1<<11	Precio Last (precio de la última transacción)
COPY_TICKS_VOLUME	1<<12	Volumen para el precio Last

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor	Descripción
COPY_TICKS_FLAGS	1<<13	Banderas de ticks
Ubicación de los datos		
COPY_TICKS_VERTICAL	1<<15	Los ticks se copian en la matriz a lo largo del eje vertical.Esto significa que los ticks recibidos en la matriz se ordenan verticalmente, es decir, los más antiguos están en la primera fila y los más recientes en la última fila de la matriz. Por defecto, al copiar en la matriz, los ticks se añaden a lo largo del eje horizontal. La bandera solo es relevante cuando se copia en una matriz.

Para saber exactamente qué datos han cambiado con el tick actual, analizaremos sus banderas:

- TICK_FLAG_BID - el tick ha cambiado el precio Bid
- TICK_FLAG_ASK - el tick ha cambiado el precio Ask
- TICK_FLAG_LAST - el tick ha cambiado el precio de la última transacción
- TICK_FLAG_VOLUME - el tick ha cambiado el volumen
- TICK_FLAG_BUY - el tick ha aparecido como resultado de una transacción de compra
- TICK_FLAG_SELL - el tick ha aparecido como resultado de una transacción de venta

Ver también

Acceso a las series temporales e indicadores, CopyTicks

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange

Obtiene los ticks de la estructura MqlTick en una matriz o vector en el intervalo de fechas indicado.El conteo de elementos desde la posición de inicio se realiza del pasado hacia el presente, es decir, el tick con el índice 0 será el más antiguo. Para analizar el tick deberemos comprobar el campo *flags*, que indica exactamente qué ha cambiado en el tick en cuestión.

```
bool  matrix::CopyTicksRange(
   string           symbol,              // nombre del símbolo
   ulong            flags,               // bandera que determina el tipo de ticks c
   ulong            from_msc,            // hora desde la que se solicitan los ticks
   ulong            to_msc               // hora hasta la que se solicitan los ticks
   );
```

Método del vector

```
bool  vector::CopyTicksRange(
   string           symbol,              // nombre del símbolo
   ulong            flags,               // bandera que determina el tipo de ticks c
   ulong            from_msc,            // hora desde la que se solicitan los ticks
   ulong            to_msc               // hora hasta la que se solicitan los ticks
   );
```

Parámetros

symbol

[in] Símbolo.

flags

[in] Combinación de banderas de la enumeración ENUM_COPY_TICKS que indican la composición de los datos solicitados. Al copiar a un vector, podemos indicar solo un valor de la enumeración ENUM_COPY_TICKS, de lo contrario, ocurrirá un error.

from_msc

[in] Hora a partir de la cual se solicitan los ticks. Se especifica en milisegundos a partir del 01.01.1970. Si el parámetro from_msc no ha sido indicado, se proporcionarán los ticks desde el inicio de la historia. Se proporcionarán los ticks con la hora >= from_msc.

to_msc

[in] Hora hasta la que se solicitan los ticks. Se especifica en milisegundos a partir del 01.01.1970. Los ticks se retornan con la hora <= to_msc. Si no se especifica el parámetro to_msc, se retornarán todos los ticks hasta el final de la historia.

Valor retornado

Retorna true en caso de éxito, de lo contrario, retornará false en caso de error. GetLastError() puede retornar los siguientes errores:

- ERR_HISTORY_TIMEOUT - el tiempo de espera para sincronizar los ticks ha expirado, la función ha ofrecido todo lo que tenía.
- ERR_HISTORY_SMALL_BUFFER - el búfer estático es demasiado pequeño, se ha ofrecido todo lo que se puede colocar en la matriz.

© 2000-2025, MetaQuotes Ltd.

- ERR_NOT_ENOUGH_MEMORY - no hay suficiente memoria para obtener en la matriz dinámica de ticks la historia del rango especificado. No se ha podido asignar la cantidad de memoria necesaria a la matriz de ticks.

Para saber exactamente qué datos han cambiado con el tick actual, analizaremos sus banderas:

- TICK_FLAG_BID – el tick ha cambiado el precio Bid
- TICK_FLAG_ASK – el tick ha cambiado el precio Ask
- TICK_FLAG_LAST – el tick ha cambiado el precio de la última transacción
- TICK_FLAG_VOLUME – el tick ha cambiado el volumen
- TICK_FLAG_BUY – el tick ha aparecido como resultado de una transacción de compra
- TICK_FLAG_SELL – el tick ha aparecido como resultado de una transacción de venta

Observación

El método CopyTicksRange() está pensado para solicitar los ticks de un rango estrictamente especificado, por ejemplo de un día concreto de la historia. A su vez, CopyTicks() permite especificar solo la fecha de inicio, por ejemplo, obtener todos los ticks desde principios de mes hasta el momento actual.

Ver también

Acceso a las series temporales e indicadores, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

Eye

La función estática crea una matriz del tamaño especificado con unidades en la diagonal especificada y ceros fuera de la diagonal. Retorna una matriz con unidades a lo largo de la diagonal y ceros en el resto.

```
static matrix matrix::Eye(
  const ulong  rows,         // número de filas
  const ulong  cols,         // número de columnas
  const int    ndiag=0       // índice de la diagonal
   );
```

Parámetros

rows

[in] Número de filas en la salida.

cols

[in] Número de columnas en la salida.

ndiag=0

[in] El índice de la diagonal: 0 (por defecto) indica la diagonal principal; el valor positivo indica la diagonal superior, el valor negativo, la diagonal inferior.

Valor retornado

Es una matriz en la que todos los elementos son iguales a cero, excepto la k-ésima diagonal, cuyos valores son iguales a uno.

Ejemplo en MQL5:

```
  matrix eye=matrix::Eye(3, 3);
  Print("eye = \n", eye);

  eye=matrix::Eye(4, 4,1);
  Print("eye = \n", eye);
  /*
   eye =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
   eye =
   [[0,1,0,0]
    [0,0,1,0]
    [0,0,0,1]
    [0,0,0,0]]
  */
```

Ejemplo en Python:

```
np.eye(3, dtype=int)
```

© 2000-2025, MetaQuotes Ltd.

```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])


 np.eye(4, k=1)
array([[0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 0., 0.]])
```

© 2000-2025, MetaQuotes Ltd.

Identity

La función estática crea una matriz unitaria de un tamaño especificado (no necesariamente cuadrada). La matriz unitaria contiene unidades en la diagonal principal y ceros fuera de la misma. La diagonal principal está formada por elementos de matriz con índices de fila y columna iguales, es decir, [0,0], [1,1],[2,2], etc. Crea una nueva matriz unitaria.

También existe el método Identity, que convierte una matriz existente en una matriz unitaria.

```
static matrix matrix::Identity(
  const ulong  rows,         // número de filas
  const ulong  cols,         // número de columnas
   );

void matrix::Identity();
```

Parámetros

rows

[in] Número de filas (y columnas) en la matriz n x n.

Valor retornado

Retorna una matriz unitaria. Una matriz unitaria es una matriz cuadrada con unidades en la diagonal principal.

Ejemplo en MQL5:

```
  matrix identity=matrix::Identity(3,3);
  Print("identity = \n", identity);
/*
   identity =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
*/
  matrix identity2(3,5);
  identity2.Identity();
  Print("identity2 = \n", identity2);
/*
   identity2 =
   [[1,0,0,0,0]
    [0,1,0,0,0]
    [0,0,1,0,0]]
*/
```

Ejemplo en Python:

```
np.identity(3)
```

© 2000-2025, MetaQuotes Ltd.

```
array([[1.,  0.,  0.],
       [0.,  1.,  0.],
       [0.,  0.,  1.]])
```

© 2000-2025, MetaQuotes Ltd.

Ones

Función estática. Crea y retorna una nueva matriz rellenada con unidades.

```
static matrix matrix::Ones(
  const ulong  rows,      // número de filas
  const ulong  cols       // número de columnas
   );

static vector vector::Ones(
  const ulong  size,      // tamaño del vector
   );
```

Parámetros

rows

[in] Número de filas.

cols

[in] Número de columnas.

Valor retornado

Retorna una nueva matriz de las filas y columnas especificadas, rellenada con unidades.

Ejemplo en MQL5:

```
  matrix ones=matrix::Ones(4, 4);
  Print("ones = \n", ones);
/*
ones =
   [[1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]]
*/
```

Ejemplo en Python:

```
np.ones((4, 1))
array([[1.],
       [1.]])
```

© 2000-2025, MetaQuotes Ltd.

Zeros

Función estática. Crea y retorna una nueva matriz rellenada con ceros.

```
static matrix matrix::Zeros(
  const ulong  rows,      // número de filas
  const ulong  cols       // número de columnas
   );

static vector vector::Zeros(
  const ulong  size,      // tamaño del vector
   );
```

Parámetros

rows

[in] Número de filas.

cols

[in] Número de columnas.

Valor retornado

Retorna una nueva matriz de las filas y columnas especificadas, rellenada con ceros.

Ejemplo en MQL5:

```
  matrix zeros=matrix::Zeros(3, 4);
  Print("zeros = \n", zeros);
/*
zeros =
   [[0,0,0,0]
    [0,0,0,0]
    [0,0,0,0]]
*/
```

Ejemplo en Python:

```
np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

© 2000-2025, MetaQuotes Ltd.

Full

Función estática. Crea y retorna una nueva matriz rellenada con el valor indicado.

```
static matrix matrix::Full(
  const ulong   rows,       // número de filas
  const ulong   cols,       // número de columnas
  const double  value       // valor para rellenar
   );

static vector vector::Full(
  const ulong   size,       // tamaño del vector
  const double  value       // valor para rellenar
   );
```

Parámetros

rows

[in] Número de filas.

cols

[in] Número de columnas.

value

[in] Valor con el que se rellenan todos los elementos de la matriz.

Valor retornado

Retorna una nueva matriz de las filas y columnas especificadas, rellenada con el valor especificado.

Ejemplo en MQL5:

```
  matrix full=matrix::Full(3,4,10);
  Print("full = \n", full);
/*
full =
   [[10,10,10,10]
    [10,10,10,10]
    [10,10,10,10]]
*/
```

Ejemplo:

```
np.full((2, 2), 10)
array([[10, 10],
       [10, 10]])
```

© 2000-2025, MetaQuotes Ltd.

Tri

Función estática. Crea una matriz con unidades en la diagonal indicada y por debajo de la misma, y ceros en el resto.

```
static matrix matrix::Tri(
  const ulong rows,         // número de filas
  const ulong cols,         // número de columnas
  const int   ndiag=0       // número de diagonales
   );
```

Parámetros

rows

[in] Número de filas en la matriz.

cols

[in] Número de columnas en la matriz.

ndiag=0

[in] Subdiagonal de la matriz que se rellenará y por debajo la cual se rellenarán las demás diagonales. k = 0 — diagonal principal, k < 0 — diagonales por debajo de la principal, k > 0 — por encima. Por defecto - 0.

Valor retornado

Matriz cuyo triángulo inferior se rellenará con unidades; todos los demás lugares se rellenarán con ceros.

Ejemplo en MQL5:

```
   matrix matrix_a=matrix::Tri(3,4,1);
   Print("Tri(3,4,1)\n",matrix_a);
   matrix_a=matrix::Tri(4,3,-1);
   Print("Tri(4,3,-1)\n",matrix_a);

/*
   Tri(3,4,1)
   [[1,1,0,0]
    [1,1,1,0]
    [1,1,1,1]]
   Tri(4,3,-1)
   [[0,0,0]
    [1,0,0]
    [1,1,0]
    [1,1,1]]
*/
```

Ejemplo:

```
np.tri(3, 5, 2, dtype=int)
array([[1, 1, 1, 0, 0],
```

© 2000-2025, MetaQuotes Ltd.

```
        [1, 1, 1, 1, 0],
        [1, 1, 1, 1, 1]])
```

© 2000-2025, MetaQuotes Ltd.

Init

Inicializa una matriz o vector.

```
void matrix::Init(
  const ulong  rows,            // número de filas
  const ulong  cols,            // número de columnas
   func_name   init_func=NULL,  // función de inicialización situada en algún ámbito
   ...         parameters
   );

void vector::Init(
  const ulong  size,            // parámetro del vector
  func_name    init_func=NULL,  // función de inicialización situada en algún ámbito
   ...         parameters
   );
```

Parámetros

rows

[in] Número de filas.

cols

[in] Número de columnas.

func_name

[in] Función de inicialización.

...

[in] Parámetros de la función de inicialización.

Valor retornado

No hay valor retornado.

Ejemplo:

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
  int size_m=3, size_k=4;
  matrix  m(size_m,size_k,MatrixArange,-2.,0.1); // primero creamos una matriz no inic
  Print("matrix m \n",m);                        // después llamamos a la función Matr
  matrixf m_float(5,5,MatrixArange,-2.f,0.1f);   // tras crear una matriz de tipo floa
  Print("matrix m_float \n",m_float);
  vector  v(size_k,VectorArange,-10.0);          // tras crear un vector, llamaremos a
  Print("vector v \n",v);
  /*
   matrix m
   [[-2,-1.9,-1.8,-1.7]
    [-1.6,-1.5,-1.399999999999999,-1.299999999999999]
    [-1.199999999999999,-1.099999999999999,-0.9999999999999992,-0.8999999999999992]]
   matrix m_float
   [[-2,-1.9,-1.8,-1.6999999,-1.5999999]
    [-1.4999999,-1.3999999,-1.2999998,-1.1999998,-1.0999998]
    [-0.99999976,-0.89999974,-0.79999971,-0.69999969,-0.59999967]
    [-0.49999967,-0.39999968,-0.29999968,-0.19999969,-0.099999689]
    [3.1292439e-07,0.10000031,0.20000032,0.30000031,0.4000003]]
   vector v
   [-10,-9,-8,-7]
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

Fill

Rellena una matriz o vector existente con el valor dado.

```
void matrix::Fill(
  const double  value      // valor para rellenar
   );

void vector::Fill(
  const double  value      // valor para rellenar
   );
```

Parámetros

value

[in] Valor con el que se rellenan todos los elementos de la matriz.

Valor retornado

No hay valor retornado. La matriz se rellena en el lugar con el valor indicado.

Ejemplo:

```
   matrix matrix_a(2,2);
   matrix_a.Fill(10);
   Print("matrix_a\n",matrix_a);

/*
  matrix_a
  [[10,10]
   [10,10]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Random

Función estática. Crea y retorna una nueva matriz o vector relleno de valores aleatorios. Los valores aleatorios se generan uniformemente dentro de un rango determinado.

```
static vector vector::Random(
  const ulong   size,       // longitud del vector
  const double  min=0.0,    // valor mínimo
  const double  max=1.0     // valor máximo
   );

static matrix matrix::Random(
  const ulong   rows,       // número de líneas
  const ulong   cols        // número de columnas
  const float   min=0.0,    // valor mínimo
  const float   max=1.0     // valor máximo
   );
```

Función para rellenar una matriz o vector existente con valores aleatorios. Los valores aleatorios se generan uniformemente dentro del rango especificado.

```
void  vector::Random(
  const double  min=0.0,    // valor mínimo
  const double  max=1.0     // valor máximo
   );

void  matrix::Random(
  const float  min=0.0,     // valor mínimo
  const float  max=1.0      // valor máximo
   );
```

Parámetros

rows

[in] Número de filas.

cols

[in] Número de columnas.

size

[in] Longitud del vector.

min=0.0

[in] Valor mínimo en la muestra generada de números aleatorios.

max=1.0

[in] Valor máximo en la muestra generada de números aleatorios.

Valor retornado

Retorna una nueva matriz a partir de las filas y columnas dadas, con valores aleatorios.

© 2000-2025, MetaQuotes Ltd.

Ejemplo

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector v1= vector::Random(6,0,4);
   Print("vector v1 \n", v1);
   /*
   vector v1
   [3.340834286841758,0.9073984578501895,3.15598511117417,0.7106475841045956,2.294010(
   */

   vector v2(4);
   v2.Random(-3,3);
   Print("vector v2 \n", v2);
   /*
   vector v2
   [-2.642032080246609,1.008521607147651,1.340623848219547,-2.789016363161853]
   */

   matrix m1 = matrix::Random(5, 5, -1, 3);
   Print("matrix m1 \n", m1);
   /*
   matrix m1
   [[1.468433369525809,1.199383023707052,0.6154667729547869,2.830012157519816,-0.06995
    [2.638591175814763,2.966658948912904,2.001837257994767,-0.0857080246987073l,0.826(
    [0.982665140643405,0.5825661973987162,-0.6615881071569371,0.4069533109215444,-0.16
    [-0.5791836119208447,-0.9332020478167575,2.801898707589577,-0.09887318478405316,-(
    [-0.1069569017521471,-0.04576879345014895,0.6320100840255642,-0.3805729270401813,(
   */

   matrix m2(3, 3);
   m2.Random(-2, 2);
   Print("matrix m2 \n", m2);
   /*
   matrix m2
   [[-1.928191291151613,-1.222873445419332,-1.965333905062949]
    [-1.077723600686929,1.053317863273741,-0.1786824246353196]
    [-1.965241401448204,1.620956906029246,-1.696386492740453]]
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Manipulaciones sobre matrices y vectores

Métodos para realizar operaciones básicas con matrices: rellenar, copiar, obtener una parte de una matriz, transponer, dividir y clasificar.

También existen varios métodos para trabajar con las filas y columnas de una matriz.

Función	Acción
HasNan	Retorna el número de valores NaN en una matriz/vector
Transpose	Gira o reorganiza los ejes de una matriz, retorna una matriz modificada
TransposeConjugate	Transposición de una matriz compleja con conjugación. Desarrolla o reorganiza los ejes de la matriz alterando el signo de la parte imaginaria de los números complejos, devolviendo la matriz modificada
TriL	Retorna una copia de la matriz, reemplazando todos los elementos por encima de la k-ésima diagonal con ceros. Matriz triangular inferior
TriU	Retorna una copia de la matriz, reemplazando todos los elementos por debajo de la k-ésima diagonal con ceros. Matriz triangular superior
Diag	Extrae una diagonal o crea una matriz diagonal
Row	Devuelve un vector de filas. Escribe el vector en la fila especificada
Col	Retorna un vector de columna. Escribe un vector en la columna especificada
Copy	Retorna la copia de una matriz/vector dado
Concat	Concatena 2 submatrices en una sola matriz. Concatena 2 vectores en uno.
Compare	Compara los elementos de dos matrices/vectores con una precisión determinada
CompareByDigits	Compara si los elementos de dos matrices/vectores coinciden con una precisión de dígitos significativos
CompareByDigits	Ejecuta una comparación absoluta de dos matrices, transformando sus filas secuenciales en vectores unidimensionales
Flat	Permite direccionar hacia un elemento de la matriz a través de un único índice en lugar de dos
Clip	Limita los elementos de una matriz/vector a un rango especificado de valores válidos
Reshape	Cambia la forma de una matriz sin cambiar sus datos
Resize	Retorna una nueva matriz de la forma y el tamaño especificados

© 2000-2025, MetaQuotes Ltd.

Función	Acción
SwapRows	Intercambia las filas de la matriz
SwapCols	Intercambia las columnas de la matriz
Split	Partición de una matriz en varias submatrices
Hsplit	Partición horizontal de una matriz en varias submatrices. Lo mismo que Split con axis=0
Vsplit	Partición vertical de una matriz en varias submatrices. Los mismo que Split c axis=1
ArgSort	Realiza la clasificación indirecta en una matriz o vector.
Sort	Clasifica una matriz o un vector según su lugar.

© 2000-2025, MetaQuotes Ltd.

HasNan

Retorna el número de valores NaN en una matriz/vector.

```
ulong vector::HasNan();

ulong matrix::HasNan();
```

Valor retornado

Número de elementos de una matriz/vector que contienen un valor NaN.

Observación

Los métodos Compare y CompareByDigits, al comparar el par correspondiente de elementos que tienen valores NaN, considera que estos elementos son iguales, mientras que en una comparación normal con coma flotante NaN != NaN.

Ejemplo:

```
void OnStart(void)
  {
   double x=sqrt(-1);

   Print("single: ",x==x);

   vector<double> v1={x};
   vector<double> v2={x};

   Print("vector: ", v1.Compare(v2,0)==0);
  }

/* Resultado:

 single: false
 vector: true
*/
```

Ver también

MathClassify, Compare, CompareByDigits

© 2000-2025, MetaQuotes Ltd.

Transpose

Transposición de matrices. Gira o reorganiza los ejes de una matriz, retorna una matriz modificada.

```
matrix matrix::Transpose()
```

Valor retornado

Matriz transpuesta.

Algoritmo sencillo de transposición de matrices en MQL5:

```
matrix MatrixTranspose(const matrix& matrix_a)
  {
   matrix matrix_c(matrix_a.Cols(),matrix_a.Rows());

   for(ulong i=0; i<matrix_c.Rows(); i++)
      for(ulong j=0; j<matrix_c.Cols(); j++)
         matrix_c[i][j]=matrix_a[j][i];

   return(matrix_c);
  }
```

Ejemplo en MQL5:

```
  matrix a= {{0, 1, 2}, {3, 4, 5}};
  Print("matrix a \n", a);
  Print("a.Transpose() \n", a.Transpose());

  /*
  matrix a
  [[0,1,2]
   [3,4,5]]
  a.Transpose()
  [[0,3]
   [1,4]
   [2,5]]
  */
```

Ejemplo en Python:

```
import numpy as np

a = np.arange(6).reshape((2,3))
print("a \n",a)
print("np.transpose(a) \n",np.transpose(a))
```

© 2000-2025, MetaQuotes Ltd.

```
a
 [[0 1 2]
 [3 4 5]]
np.transpose(a)
 [[0 3]
 [1 4]
 [2 5]]
```

© 2000-2025, MetaQuotes Ltd.

TransposeConjugate

Transposición de una matriz compleja con conjugación. Desarrolla o reorganiza los ejes de la matriz alterando el signo de la parte imaginaria de los números complejos, devolviendo la matriz modificada.

```
matrix matrix::TransposeConjugate()
```

Valor devuelto

Matriz transpuesta y conjugada compleja.

Algoritmo simple de transposición de una matriz compleja con conjugación — explicación del método

```
//--- Función de transposición de una matriz compleja con conjugación
matrix MatrixTranspose(const matrix& matrix_a) {
    //--- creamos una nueva matriz matrix_c con dimensiones invertidas en relación a l
    matrix matrix_c(matrix_a.Cols(), matrix_a.Rows());

    //--- recorremos todas las filas de la nueva matriz
    for (ulong i = 0; i < matrix_c.Rows(); i++) {
        //--- recorremos todas las columnas de la nueva matriz
        for (ulong j = 0; j < matrix_c.Cols(); j++) {
            //--- transferimos la parte real del elemento, transponiendo el índice
            matrix_c[i][j].real = matrix_a[j][i].real;

            //--- transferimos la parte imaginaria del elemento, cambiando el signo (c
            matrix_c[i][j].imag = -matrix_a[j][i].imag;
        }
    }

    //--- devolvemos la matriz transpuesta y conjugada
    return (matrix_c);
}
```

© 2000-2025, MetaQuotes Ltd.

TriL

Retorna una copia de la matriz, reemplazando todos los elementos por encima de la k-ésima diagonal con ceros. Matriz triangular inferior.

```
matrix matrix::Tril(
  const int      ndiag=0      // índice de la diagonal
   );
```

Parámetros

ndiag=0

[in] Diagonal por encima de la cual los elementos se sustituyen por ceros. ndiag = 0 (por defecto) — diagonal principal, ndiag < 0 — por debajo, y ndiag > 0 — por encima de la diagonal principal.

Valor retornado

Matriz cuyo triángulo inferior se rellenará con unidades; todos los demás lugares se rellenarán con ceros.

Ejemplo en MQL5:

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriL(-1);
   Print("matrix b \n",b);

  /*
  matrix_c
  [[0,0,0]
  [4,0,0]
  [7,8,0]
  [10,11,12]]
  */
```

Ejemplo en Python:

```
import numpy as np

a=np.tril([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)

[[ 0  0  0]
 [ 4  0  0]
 [ 7  8  0]
 [10 11 12]]
```

© 2000-2025, MetaQuotes Ltd.

TriU

Retorna una copia de la matriz, reemplazando todos los elementos por debajo de la k-ésima diagonal con ceros. Matriz triangular superior.

```
matrix matrix::Triu(
  const int      ndiag=0      // índice de la diagonal
   );
```

Parámetros

ndiag=0

[in] Diagonal por debajo de la cual los elementos se sustituyen por ceros. ndiag = 0 (por defecto) — diagonal principal, ndiag < 0 — por debajo, y ndiag > 0 — por encima de la diagonal principal.

Ejemplo en MQL5:

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriU(-1);
   Print("matrix b \n",b);

  /*
  matrix b
  [[1,2,3]
   [4,5,6]
   [0,8,9]
   [0,0,12]]
  */
```

Ejemplo en Python:

```
import numpy as np

a=np.triu([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)
print(a)

[[ 1  2  3]
 [ 4  5  6]
 [ 0  8  9]
 [ 0  0 12]]
```

© 2000-2025, MetaQuotes Ltd.

Diag

Extrae una diagonal o crea una matriz diagonal.

```
vector matrix::Diag(
  const int     ndiag=0      // número de la diagonal
   );

void matrix::Diag(
  const vector  v,           // vector de diagonal
  const int     ndiag=0      // número de la diagonal
   );
```

Parámetros

v

[in] Vector cuyos elementos deben inscribirse en la diagonal correspondiente (ndiag=0 — diagonal principal).

ndiag=0

[in] Diagonal. Por defecto - 0. Para las diagonales por encima de la principal, ndiag>0, para las diagonales por debajo, ndiag<0.

Observación

Podemos establecer una diagonal para matrices no distribuidas (sin dimensiones). En este caso, se creará una matriz nula con el tamaño correspondiente al vector diagonal y, a continuación, los valores de los elementos del vector se insertarán en la diagonal correspondiente. Si establecemos una diagonal en una matriz existente, las dimensiones de la matriz no cambiarán y los valores de los elementos de la matriz fuera del vector de diagonal tampoco cambiarán.

Ejemplo:

```
   vector v1={1,2,3};
   matrix m1;
   m1.Diag(v1);
   Print("m1\n",m1);
   matrix m2;
   m2.Diag(v1,-1);
   Print("m2\n",m2);
   matrix m3;
   m3.Diag(v1,1);
   Print("m3\n",m3);
   matrix m4=matrix::Full(4,5,9);
   m4.Diag(v1,1);
   Print("m4\n",m4);

   Print("diag -1 - ",m4.Diag(-1));
```

© 2000-2025, MetaQuotes Ltd.

```
 Print("diag 0 - ",m4.Diag());
 Print("diag 1 - ",m4.Diag(1));

/*

m1
[[1,0,0]
[0,2,0]
[0,0,3]]
m2
[[0,0,0]
[1,0,0]
[0,2,0]
[0,0,3]]
m3
[[0,1,0,0]
[0,0,2,0]
[0,0,0,3]]
m4
[[9,1,9,9,9]
[9,9,2,9,9]
[9,9,9,3,9]
[9,9,9,9,9]]
diag -1 - [9,9,9]
diag 0 - [9,9,9,9]
diag 1 - [1,2,3,9]
*/
```

© 2000-2025, MetaQuotes Ltd.

Row

Devuelve un vector de filas. Escribe el vector en la fila especificada.

```
vector matrix::Row(
  const ulong   nrow      // número de fila
   );

void matrix::Row(
  const vector  v,        // vector de la fila
  const ulong   nrow      // número de fila
   );
```

Parámetros

nrow

[in] Número de fila.

Valor retornado

Vector.

Observación

Podemos establecer una fila para las matrices no especificadas (sin dimensiones). En este caso, se creará una matriz cero de tamaño número de fila + 1 x tamaño del vector, luego de eso, los valores de los elementos del vector se colocan en la fila correspondiente. Si la fila se establece en una matriz ya existente, las dimensiones de la matriz no cambiarán y los valores de los elementos de la matriz fuera del vector de fila tampoco cambiarán.

Ejemplo:

```
   vector v1={1,2,3};
   matrix m1;
   m1.Row(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,7);
   m2.Row(v1,2);
   Print("m2\n",m2);

   Print("row 1 - ",m2.Row(1));
   Print("row 2 - ",m2.Row(2));

  /*
  m1
  [[0,0,0]
  [1,2,3]]
  m2
  [[7,7,7,7,7]
```

© 2000-2025, MetaQuotes Ltd.

```
  [7,7,7,7,7]
  [1,2,3,7,7]
  [7,7,7,7,7]]
  row 1 - [7,7,7,7,7]
  row 2 - [1,2,3,7,7]
  */
```

© 2000-2025, MetaQuotes Ltd.

Col

Retorna un vector de columna. Escribe un vector en la columna especificada.

```
vector matrix::Col(
  const ulong   ncol      // número de columna
   );

void matrix::Col(
  const vector  v,        // vector de columna
  const ulong   ncol      // número de columna
   );
```

Parámetros

ncol

[in] Número de columna.

Valor retornado

Vector.

Observación

Podemos establecer una columna para las matrices no especificadas (sin dimensiones). En este caso, se creará una matriz nula con un tamaño de vector igual al vector x el número de columna+1; después de ello, los valores de los elementos del vector se insertarán en la columna correspondiente. Si la columna se coloca en una matriz existente, las dimensiones de la matriz no cambiarán y los valores de los elementos de la matriz fuera del vector de columna tampoco cambiarán.

Ejemplo:

```
   vector v1={1,2,3};
   matrix m1;
   m1.Col(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,8);
   m2.Col(v1,2);
   Print("m2\n",m2);

   Print("col 1 - ",m2.Col(1));
   Print("col 2 - ",m2.Col(2));

  /*
  m1
  [[0,1]
  [0,2]
  [0,3]]
  m2
  [[8,8,1,8,8]
```

© 2000-2025, MetaQuotes Ltd.

```
   [8,8,2,8,8]
   [8,8,3,8,8]
   [8,8,8,8,8]]
   col 1 - [8,8,8,8]
   col 2 - [1,2,3,8]
   */
```

© 2000-2025, MetaQuotes Ltd.

Copy

Creamos una copia de la matriz/vector dado.

```
bool matrix::Copy(
  const matrix&  a     // matriz a copiar
   );
bool vector::Copy(
  const vector&  v     // vector a copiar
   );
```

Parámetros

v

[in] Matriz o vector a copiar.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Ejemplo en MQL5:

```
  matrix a=matrix::Eye(3, 4);
  matrix b;
  b.Copy(a);
  matrix c=a;
  Print("matrix b \n", b);
  Print("matrix_c \n", c);

  /*
  /*
  matrix b
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  matrix_c
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  */
  */
```

Ejemplo en Python:

```
import numpy as np

a = np.eye(3,4)
print('a \n',a)
b = a
```

© 2000-2025, MetaQuotes Ltd.

```
print('b \n',b)
c = np.copy(a)
print('c \n',c)

a
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
b
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
c
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
```

Ver también

Assign

© 2000-2025, MetaQuotes Ltd.

Concat

Concatena 2 submatrices en una sola matriz. Concatena 2 vectores en uno.

```
vector vector::Concat(
  const vector&  second_part      // segundo vector a concatenar
   );

vector matrix::Concat(
  const matrix&  second_part      // segunda matriz a concatenar
   );

matrix matrix::Split(
  const matrix&  second_part,     // segunda matriz a concatenar
  const int      axis             // eje
   );
```

Parámetros

second_part

[in] Segundo vector o matriz a concatenar. Si se concatenan matrices a lo largo de cualquier eje, entonces los tamaños de las matrices deberán ser consistentes a lo largo de ese eje.

axis

[in] Eje. 0 - eje horizontal, 1 - eje vertical.

Valor retornado

Retorna un vector si los vectores o matrices se concatenan sin un parámetro de eje, o las matrices se concatenan a lo largo de un eje horizontal o vertical.

Ejemplo

```
   vector vector_a={1,2,3,4};
   vector vector_b={5,6,7};
   vector vector_c=vector_a.Concat(vector_b);
   Print("vector_c=",vector_c);

   matrix matrix_a={{1,2,3},{4,5,6}};
   matrix matrix_b={{7,8,9},{10,11,12}};
   vector_c=matrix_a.Concat(matrix_b);
   Print("vector_c=",vector_c);

   matrix matrix_c0=matrix_a.Concat(matrix_b,0);
   Print("matrix_c0=\n",matrix_c0);

   matrix matrix_c1=matrix_a.Concat(matrix_b,1);
   Print("matrix_c1=\n",matrix_c1);

  /*
```

© 2000-2025, MetaQuotes Ltd.

```
vector_c=[1,2,3,4,5,6,7]
vector_c=[1,2,3,4,5,6,7,8,9,10,11,12]
matrix_c0=
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
matrix_c1=
[[1,2,3,7,8,9]
 [4,5,6,10,11,12]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Compare

Compara los elementos de dos matrices/vectores con una precisión determinada.

```
ulong vector::Compare(
  const vector& vec,          // vector para la comparación
  const double  epsilon       // precisión
   );

ulong matrix::Compare(
  const matrix& mat,          // matriz para la comparación
  const double  epsilon       // precisión
   );
```

Parámetros

vector_b

[in] Vector para la comparación.

vector_b

[in] Vector para la comparación.

epsilon

[in] Precisión.

Valor retornado

Retorna el número de elementos no coincidentes en las matrices o vectores comparados, es decir, tendremos 0 si las matrices son iguales, o un número mayor que 0 en caso contrario.

Para las matrices/vectores complejos, la comparación consiste en estimar la distancia entre números complejos. La distancia se calcula de acuerdo con la fórmula siguiente sqrt(pow(r1-r2, 2) + pow(i1-i2, 2) y supone un número real que ya se puede comparar con épsilon.

Observación

Las operaciones de comparación == o != realizan una comparación por elementos exacta. Como sabemos, la comparación exacta de números reales tiene un uso limitado, por lo que hemos añadido el método de comparación con épsilon. Hay casos en los que dentro de una misma matriz hay elementos en el rango, por ejemplo, de 1e-20 hasta 1e+20. Para ello, se contempla la comparación por elementos, teniendo en cuenta los dígitos significativos.

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4}};
   matrix matrix_i=matrix::Identity(3,3);
   matrix matrix_c=matrix_a.Inv();
   matrix matrix_check=matrix_a.MatMul(matrix_c);
```

© 2000-2025, MetaQuotes Ltd.

```
 Print("matrix_check\n",matrix_check);

 ulong errors=matrix_check.Compare(matrix::Identity(3,3),1e-15);
 Print("errors=",errors);


/*
matrix_check
[[1,0,0]
[4.440892098500626e-16,1,8.881784197001252e-16]
[4.440892098500626e-16,2.220446049250313e-16,0.9999999999999996]]
errors=0

*/
```

© 2000-2025, MetaQuotes Ltd.

CompareByDigits

Compara si los elementos de dos matrices/vectores coinciden con una precisión de dígitos significativos.

```
ulong vector::CompareByDigits(
  const vector&  vec,          // vector para comparación
  const int     digits         // número de dígitos significativos
   );

ulong matrix::CompareByDigits(
  const matrix&  mat,          // matriz para la comparación
  const int      digits        // número de dígitos significativos
   );
```

Parámetros

vector_b

[in] Vector para la comparación.

digits

[in] Número de dígitos significativos para la comparación.

epsilon

[in] Precisión de la comparación. Si dos valores difieren en el módulo en una magnitud menor que la precisión especificada, se considerarán iguales.

Valor retornado

Retorna el número de elementos no coincidentes en las matrices o vectores comparandos, es decir, 0 si las matrices son iguales; en caso contrario, el valor será superior a 0.

Observación

Las operaciones de comparación == o != realizan una comparación por elementos exacta. Como sabemos, la comparación exacta de números reales tiene un uso limitado, por lo que hemos añadido el método de comparación con épsilon. Hay casos en los que dentro de una misma matriz hay elementos en el rango, por ejemplo, de 1e-20 hasta 1e+20. Para ello, se contempla la comparación por elementos, teniendo en cuenta los dígitos significativos.

Ejemplo:

```
   int    size_m=128;
   int    size_k=256;
   matrix matrix_a(size_m,size_k);
//--- rellenamos la matriz
   double  value=0.0;
   for(int i=0; i<size_m; i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      for(int j=0; j<size_k; j++)
        {
         if(i==j)
            matrix_a[i][j]=1.0+i;
         else
           {
            value+=1.0;
            matrix_a[i][j]=value/1e+20;
           }
        }
     }
//--- obtenemos otra matriz
   matrix matrix_c = matrix_a * -1;

   ulong errors_epsilon=matrix_a.Compare(matrix_c,1e-15);
   ulong errors_digits=matrix_a.CompareByDigits(matrix_c,15);

   printf("Compare matrix %d x %d  errors_epsilon=%I64u  errors_digits=%I64u",size_m,s


  /*
  Compare matrix 128 x 256  errors_epsilon=128  errors_digits=32768
  */
```

© 2000-2025, MetaQuotes Ltd.

CompareEqual

Ejecuta una comparación absoluta de dos matrices, transformando sus filas secuenciales en vectores unidimensionales.

```
ulong vector::Compare(
  const vector& vec    // vector para comparación
   );

ulong matrix::CompareEqual(
  const matrix& mat     // matriz para comparación
   );
```

Parámetros

vec

[in] Vetor para comparación.

mat

[in] Matriz para comparación.

Descripción del método

Supongamos que tenemos dos matrices: la matriz A — matriz que llama al método — y la matriz B, pasada como parámetro. La comparación sigue los siguientes pasos:

1. Las matrices se transforman en vectores unidimensionales, concatenando sus filas secuencialmente.
2. Los vectores resultantes se comparan elemento por elemento, hasta el primer elemento divergente.
3. El método devuelve uno de los valores descritos a continuación, dependiendo del resultado de la comparación.

Valor devuelto

-1 — si un elemento de la matriz A es menor que el elemento correspondiente en la matriz B.

0 — si todos los elementos de las matrices A y B son idénticos.

1 — si un elemento de la matriz A es mayor que el elemento correspondiente en la matriz B.

Observación

Al comparar, se tienen en cuenta los elementos con valor NaN.

Los elementos NaN se tratan como iguales si están en la misma posición en ambas matrices.

El signo de NaN no se considera en la comparación.

Un elemento NaN se considera menor que cualquier otro valor numérico.

Example

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   matrix matrix_a= {{10, 3, 2}, {1, 8, 12}, {6, 5, 4}};
   matrix matrix_i=matrix::Identity(3, 3);
   matrix matrix_c=matrix_a.Inv();
   matrix matrix_check=matrix_a.MatMul(matrix_c);
   Print("matrix_check\n", matrix_check);

   ulong errors=matrix_check.Compare(matrix::Identity(3, 3), 1e-15);
   Print("errors=", errors);
   /*
   matrix_check
   [[1,0,0]
   [4.440892098500626e-16,1,8.881784197001252e-16]
   [4.440892098500626e-16,2.220446049250313e-16,0.9999999999999996]]
   errors=0
   */

//--- comparación absoluta de matrizes
   matrix<double> A = matrix_a;  // Inicialización de la matriz A
   matrix<double> B = matrix_c;  // Inicialización de la matriz B
   int result = A.CompareEqual(B);
   switch(result)
     {
      case -1:
         Print("La matriz A es menor que la matriz B.");
         break;
      case 0:
         Print("Las matrices A y B son idénticas.");
         break;
      case 1:
         Print("La matriz A es mayor que la matriz B.");
         break;
      case -2:
         Print("Error! La matriz A no fue inicializada.");
         break;
      case 2:
         Print("Error! La matriz B no fue inicializada.");
         break;
      case -3:
         Print("Error! El tamaño de la matriz A es menor que el tamaño de la matriz B'
         break;
      case 3:
         Print("Error! El tamaño de la matriz A es mayor que el tamaño de la matriz B'
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
      default:
         Print("Error! Error desconocido");
         break;
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

Flat

Permite direccionar hacia un elemento de la matriz a través de un único índice en lugar de dos.

```
bool matrix::Flat(
  const ulong    index,      //
  const double    value      // valor establecido
   );

double matrix::Flat(
  const ulong    index,      //
   );
```

Parámetros

index

[in] Índice flat

value

[in] Valor a establecer según el índice especificado.

Valor retornado

Valor según el índice especificado.

Observación

Para la matriz matrix mat(3,3), los accesos se pueden escribir así:

- para la lectura — x=mat.Flat(4), lo que equivale en escritura a x=mat[1][1]
- para la escritura — mat.Flat(5, 42), lo que equivale en escritura a mat[1][2]=42

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);
   ulong arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));
   matrix_a.Flat(arg_max,0);
   arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));


  /*
  matrix_a
  [[10,3,2]
   [1,8,12]
   [6,5,4]
   [7,11,9]]
```

© 2000-2025, MetaQuotes Ltd.

```
max_value=12.0
max_value=11.0
*/
```

© 2000-2025, MetaQuotes Ltd.

Clip

Limita los elementos de una matriz/vector a un rango especificado de valores válidos.

```
bool matrix::Clip(
  const double  min_value,     // valor mínimo
  const double  max_value      // valor máximo
   );
bool vector::Clip(
  const double  min_value,     // valor mínimo
  const double  max_value      // valor máximo
   );
```

Parámetros

min_value

[in] Valor mínimo.

max_value

[in] Valor máximo.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

La matriz (o vector) se procesa en el lugar. No se crea ninguna copia.

Ejemplo:

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   bool res=matrix_a.Clip(4,8);
   Print("matrix_a\n",matrix_a);

  /*
  matrix_a
  [[4,4,4]
   [4,5,6]
   [7,8,8]
   [8,8,8]]
  */
```

© 2000-2025, MetaQuotes Ltd.

Reshape

Cambia la forma de una matriz sin cambiar sus datos.

```
void  Reshape(
  const ulong  rows,      // nuevo número de filas
  const ulong  cols       // nuevo número de columnas
   );
```

Parámetros

rows

[in] Nuevo número de filas.

cols

[in] Nuevo número de columnas.

Observación

La matriz se procesa según el lugar. No se crea ninguna copia. Podemos establecer las dimensiones que deseemos, es decir, rows_new*cols_new!=rows_old*cols_old. Al aumentar el búfer de la matriz, los valores "sobrantes" no estarán definidos.

Ejemplo:

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};

   Print("matrix_a\n",matrix_a);
   matrix_a.Reshape(2,6);
   Print("Reshape(2,6)\n",matrix_a);
   matrix_a.Reshape(3,5);
   Print("Reshape(3,5)\n",matrix_a);
   matrix_a.Reshape(2,4);
   Print("Reshape(2,4)\n",matrix_a);

  /*
  matrix_a
  [[1,2,3]
   [4,5,6]
   [7,8,9]
   [10,11,12]]
  Reshape(2,6)
  [[1,2,3,4,5,6]
   [7,8,9,10,11,12]]
  Reshape(3,5)
  [[1,2,3,4,5]
   [6,7,8,9,10]
   [11,12,0,3,0]]
  Reshape(2,4)
  [[1,2,3,4]
```

© 2000-2025, MetaQuotes Ltd.

```
   [5,6,7,8]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Resize

Retorna una nueva matriz de la forma y el tamaño especificados.

```
bool matrix::Resize(
  const ulong  rows,     //
  const ulong  cols,     // nuevo número de columnas
  const ulong  reserve=0 // número de elementos reservados
   );

bool vector::Resize(
  const ulong  size,     // nuevo tamaño
  const ulong  reserve=0 // número de elementos reservados
   );
```

Parámetros

rows

[in] Nuevo número de filas.

cols

[in] Nuevo número de columnas.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

La matriz (o vector) se procesa en el lugar. No se crea ninguna copia. Podemos establecer las dimensiones que deseemos, es decir, rows_new*cols_new!=rows_old*cols_old. A diferencia de Reshape, la matriz se procesa línea por línea. Si aumenta el número de columnas, los valores de las columnas "sobrantes" no estarán definidos. Al aumentar el número de filas, los valores de los elementos de las nuevas filas no estarán definidos. Al reducir el número de columnas, cada fila de la matriz se truncará.

Ejemplo:

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);
   matrix_a.Resize(2,6);
   Print("Ressize(2,6)\n",matrix_a);
   matrix_a.Resize(3,5);
   Print("Resize(3,5)\n",matrix_a);
   matrix_a.Resize(2,4);
   Print("Resize(2,4)\n",matrix_a);

   /*
   matrix_a
```

© 2000-2025, MetaQuotes Ltd.

```
   [[1,2,3]
    [4,5,6]
    [7,8,9]
    [10,11,12]]
   Ressize(2,6)
   [[1,2,3,4,5,6]
    [4,5,6,10,11,12]]
   Resize(3,5)
   [[1,2,3,4,5]
    [4,5,6,10,11]
    [11,12,3,8,8]]
   Resize(2,4)
   [[1,2,3,4]
    [4,5,6,10]]
   */
```

© 2000-2025, MetaQuotes Ltd.

Set

Establece el valor para el elemento del vector según el índice especificado.

```
bool vector::Set(
  ulong   index,     // índice del elemento
  double  value      // valor
   );
```

Parámetros

index

[in] Índice del elemento para el que se debe establecer el valor.

value

[in] Valor.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

El método Set hace lo mismo que la asignación del valor a través de corchetes, a saber: *vector[index]=value*. Se ha añadido para transferir con mayor comodidad el código desde los lenguajes donde se utiliza un registro de este tipo. El ejemplo a continuación muestra las dos opciones para rellenar el vector con los valores según el índice especificado.

Ejemplo:

```
void OnStart()
  {
//---
   vector v1(10, VectorAssignValues);
   Print("v1 = ", v1);

   vector v2(10, VectorSetValues);
   Print("v2 = ", v2);
  }
 /* Resultado
  v1 = [1,2,4,8,16,32,64,128,256,512]
  v2 = [1,2,4,8,16,32,64,128,256,512]
  */
//+------------------------------------------------------------------+
//| Rellena el vector con grados de un número usando una operación de asignación
//+------------------------------------------------------------------+
void VectorAssignValues(vector& v, double initial=1)
  {
   double value=initial;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(ulong k=0; k<v.Size(); k++)
     {
      v[k]=value;
      value*=2;
     }
  }
//+------------------------------------------------------------------+
//| Rellena el vector con grados de un número usando el método Set
//+------------------------------------------------------------------+
void VectorSetValues(vector& v, double initial=1)
  {
   double value=initial;
   for(ulong k=0; k<v.Size(); k++)
     {
      v.Set(k, value);
      value*=2;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SwapRows

Intercambia las filas de la matriz.

```
bool matrix::SwapRows(
  const ulong  row1,     // índice de la primera fila
  const ulong  row2      // índice de la segunda fila
   );
```

Parámetros

row1

[in] Índice de la primera fila.

row2

[in] Índice de la segunda fila.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Ejemplo:

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapRows(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapRows(0,3);
   matrix matrix_c1=matrix_p.MatMul(matrix_a);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  matrix_c1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  */
```

© 2000-2025, MetaQuotes Ltd.

SwapCols

Intercambia las columnas de la matriz.

```
bool matrix::SwapCols(
  const ulong  col1,     // índice de la primera columna
  const ulong  col2      // índice de la segunda columna
   );
```

Parámetros

col1

[in] Índice de la primera columna.

col2

[in] Índice de la segunda columna.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Ejemplo:

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapCols(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapCols(0,3);
   matrix matrix_c1=matrix_a.MatMul(matrix_p);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  matrix_c1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  */
```

© 2000-2025, MetaQuotes Ltd.

Split

Partición de una matriz en varias submatrices.

```
bool matrix::Split(
  const ulong  parts,      // número de submatrices
  const int    axis,       // eje
  matrix&      splitted[]  // array de submatrices obtenidas
   );

void matrix::Split(
  const ulong& parts[],    // dimensiones de las submatrices
  const int    axis,       // eje
  matrix&      splitted[]  // array de submatrices obtenidas
   );
```

Parámetros

parts

[in] Número de submatrices en el que se debe dividir la matriz.

axis

[in] Eje. 0 - eje horizontal, 1 - eje vertical.

splitted

[out] Array de submatrices obtenidas.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

Si se especifica el número de submatrices, se obtendrán submatrices del mismo tamaño. Es decir, el tamaño correspondiente de la matriz (0 - número de filas, 1 - número de columnas) deberá ser divisible por el número "parts" sin resto. Se pueden obtener matrices de diferentes tamaños utilizando un array de tamaños de submatrices. Los elementos del array de tamaños se utilizarán hasta dividir toda la matriz. Si el array de tamaños se ha agotado y la matriz aún no está completamente dividida, el resto sin dividir será la última submatriz.

Ejemplo:

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
   ulong  parts[]={2,2};
```

© 2000-2025, MetaQuotes Ltd.

```
 bool res=matrix_a.Split(2,0,splitted);
 Print(res,"  ",GetLastError());
 ResetLastError();
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Split(2,1,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Split(parts,0,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

/*
false  4003
true  0
splitted 0
[[1,2,3]
 [7,8,9]
 [13,14,15]
 [19,20,21]
 [25,26,27]]
splitted 1
[[4,5,6]
 [10,11,12]
 [16,17,18]
 [22,23,24]
 [28,29,30]]
true  0
splitted 0
[[1,2,3,4,5,6]
 [7,8,9,10,11,12]]
splitted 1
[[13,14,15,16,17,18]
 [19,20,21,22,23,24]]
splitted 2
[[25,26,27,28,29,30]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Hsplit

Partición horizontal de una matriz en varias submatrices. Lo mismo que Split con axis=0

```
bool matrix::Hsplit(
  const ulong  parts,      // número de submatrices
  matrix&      splitted[]  // array de submatrices obtenidas
   );

void matrix::Hsplit(
  const ulong& parts[],    // dimensiones de las submatrices
  matrix&      splitted[]  // array de submatrices obtenidas
   );
```

Parámetros

parts

[in] Número de submatrices en el que se debe dividir la matriz.

splitted

[out] Array de submatrices obtenidas.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

Si se especifica el número de submatrices, se obtendrán submatrices del mismo tamaño. Es decir, el número de filas deberá dividirse por el número "parts" sin resto. Se pueden obtener matrices de diferentes tamaños utilizando un array de tamaños de submatrices. Los elementos del array de tamaños se utilizarán hasta dividir toda la matriz. Si el array de tamaños se ha agotado y la matriz aún no está completamente dividida, el resto sin dividir será la última submatriz.

Ejemplo:

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
   ulong  parts[]={2,4};

   bool res=matrix_a.Hsplit(2,splitted);
   Print(res,"  ",GetLastError());
   ResetLastError();
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);
```

© 2000-2025, MetaQuotes Ltd.

```
 res=matrix_a.Hsplit(5,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Hsplit(parts,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);


/*
false  4003
true  0
splitted 0
[[1,2,3,4,5,6]]
splitted 1
[[7,8,9,10,11,12]]
splitted 2
[[13,14,15,16,17,18]]
splitted 3
[[19,20,21,22,23,24]]
splitted 4
[[25,26,27,28,29,30]]
true  0
splitted 0
[[1,2,3,4,5,6]
[7,8,9,10,11,12]]
splitted 1
[[13,14,15,16,17,18]
[19,20,21,22,23,24]
[25,26,27,28,29,30]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Vsplit

Partición vertical de una matriz en varias submatrices. Lo mismo que Split con axis=1

```
bool matrix::Vsplit(
  const ulong  parts,      // número de submatrices
  matrix&      splitted[]  // array de submatrices obtenidas
   );

void matrix::Vsplit(
  const ulong& parts[],    // dimensiones de las submatrices
  matrix&      splitted[]  // array de submatrices obtenidas
   );
```

Parámetros

parts

[in] Número de submatrices en el que se debe dividir la matriz.

splitted

[out] Array de submatrices obtenidas.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

Si se especifica el número de submatrices, se obtendrán submatrices del mismo tamaño. Es decir, el número de columnas deberá dividirse por el número "parts" sin resto. Se pueden obtener matrices de diferentes tamaños utilizando un array de tamaños de submatrices. Los elementos del array de tamaños se utilizarán hasta dividir toda la matriz. Si el array de tamaños se ha agotado y la matriz aún no está completamente dividida, el resto sin dividir será la última submatriz.

Ejemplo:

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18}};
   matrix splitted[];
   ulong  parts[]={2,3};

   matrix_a.Vsplit(2,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   matrix_a.Vsplit(3,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);
```

© 2000-2025, MetaQuotes Ltd.

```
 matrix_a.Vsplit(parts,splitted);
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);


/*
   splitted 0
   [[1,2,3]
    [7,8,9]
    [13,14,15]]
   splitted 1
   [[4,5,6]
    [10,11,12]
    [16,17,18]]

   splitted 0
   [[1,2]
    [7,8]
    [13,14]]
   splitted 1
   [[3,4]
    [9,10]
    [15,16]]
   splitted 2
   [[5,6]
    [11,12]
    [17,18]]

   splitted 0
   [[1,2]
    [7,8]
    [13,14]]
   splitted 1
   [[3,4,5]
    [9,10,11]
    [15,16,17]]
   splitted 2
   [[6]
    [12]
    [18]]

*/
```

© 2000-2025, MetaQuotes Ltd.

ArgSort

Clasificación indirecta de una matriz o un vector.

```
vector vector::Sort(
  func_name  compare_func=NULL,  // función de comparación
  T          context             // parámetro para la función de clasificación de usua
   );

matrix matrix::Sort(
  func_name  compare_func=NULL   // función de comparación
  T          context             // parámetro para la función de clasificación de usua
   );

matrix matrix::Sort(
  const int  axis,               // eje para la clasificación
  func_name  compare_func=NULL   // función de comparación
  T          context             // parámetro para la función de clasificación de usua
   );
```

Parámetros

axis

[in] Eje para la clasificación: 0 — horizontal, 1 — vertical.

func_name

[in] Comparador. Podemos especificar uno de los valores de la enumeración ENUM_SORT_MODE o nuestra propia función de comparación. Si no especificamos ninguna función, se utilizará la clasificación ascendente.

La función de comparación personalizada puede ser de dos tipos:

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Aquí T será el tipo de matriz o vector, mientras que TContex será el tipo de la variable context transmitida como parámetro opcional al método Sort.

context

[in] Parámetro opcional que se puede transmitir a la función de clasificación personalizada.

Valor retornado

Vector o matriz con los índices de los elementos clasificados. Por ejemplo, el resultado [4,2,0,1,3] indica que la posición cero del vector clasificado deberá tener un elemento con el índice 4, la primera posición deberá tener un elemento con el índice 2, etc.

© 2000-2025, MetaQuotes Ltd.

Sort

Clasifica una matriz o un vector según su lugar.

```
void vector::Sort(
  func_name  compare_func=NULL,  // función de comparación
  T          context             // parámetro para la función de clasificación de usua
   );

void matrix::Sort(
  func_name  compare_func=NULL   // función de comparación
  T          context             // parámetro para la función de clasificación de usua
   );

void matrix::Sort(
  const int  axis,               // eje para la clasificación
  func_name  compare_func=NULL   // función de comparación
  T          context             // parámetro para la función de clasificación de usua
   );
```

Parámetros

axis

[in] Eje para la clasificación: 0 — horizontal, 1 — vertical.

func_name

[in] Comparador. Podemos especificar uno de los valores de la enumeración ENUM_SORT_MODE o nuestra propia función de comparación. Si no especificamos ninguna función, se utilizará la clasificación ascendente.

La función de comparación personalizada puede ser de dos tipos:

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Aquí T será el tipo de matriz o vector, mientras que TContex será el tipo de la variable context transmitida como parámetro opcional al método Sort.

context

[in] Parámetro opcional que se puede transmitir a la función de clasificación personalizada.

Valor retornado

No. La clasificación se realiza según el lugar: se aplica a los datos de la matriz/vector para los que se llama al método Sort.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Función de clasificación                                         |
//+------------------------------------------------------------------+
int MyDoubleComparator(double x1,double x2,int sort_mode=0)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   int res=x1<x2 ? -1 : (x1>x2 ? 1 : 0);
   return(sort_mode==0 ? res : -res);
  }
//+------------------------------------------------------------------+
//| Función de inicio del script                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- rellando el vector
   vector v(100);
   //--- clasificación por orden ascendente
   v.Sort(MyDoubleComparator);   // aquí se utiliza un parámetro adicional con un valo
   Print(v);
   // clasificación por orden descendente
   v.Sort(MyDoubleComparator,1); // aquí el usuario especificará explícitamente el par
   Print(v);
  }
```

© 2000-2025, MetaQuotes Ltd.

Operaciones matemáticas sobre matrices y vectores

Las operaciones matemáticas (suma, resta, multiplicación y división) se pueden realizar en matrices y vectores miembro a miembro.

Las funciones matemáticas se diseñaron originalmente para realizar operaciones matemáticas en valores escalares. Ahora, la mayoría de estas funciones también se pueden usar con los nuevos tipos de datos — matrices y vectores — MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, MathTanh. En este caso, la matriz o vector se procesa por miembros. Ejemplo:

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

En el caso de MathMod y MathPow, como segundo parámetro, podremos usar tanto un escalar como una matriz o vector del tamaño correspondiente

Vamos a ver un ejemplo en el que mostraremos cómo calcular la desviación estándar usando funciones matemáticas sobre un vector.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- usamos una función de inicialización para rellenar el vector
  vector r(10, ArrayRandom); // matriz de números aleatorios de 0 a 1
//--- calculamos el valor medio
  double avr=r.Mean();       // valor medio del array
  vector d=r-avr;            // calculamos el array de desviaciones respecto al valor
  Print("avr(r)=", avr);
  Print("r=", r);
  Print("d=", d);
  vector s2=MathPow(d, 2);   // array de cuadrados de las desviaciones
  double sum=s2.Sum();       // suma de los cuadrados de las desviaciones
//--- calcularemos la desviación estándar de 2 maneras
  double std=MathSqrt(sum/r.Size());
  Print(" std(r)=", std);
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("r.Std()=", r.Std());
  }
/*
   avr(r)=0.5300302133243813
   r=[0.8346201971495713,0.8031556138798182,0.6696676534318063,0.05386516922513505,0.54
   d=[0.30458998382519,0.2731254005554369,0.1396374401074251,-0.4761650440992462,0.0190
    std(r)=0.2838269732183663
   r.Std()=0.2838269732183663
*/
//+------------------------------------------------------------------+
//| Rellena el vector con valores aleatorios                         |
//+------------------------------------------------------------------+
void ArrayRandom(vector& v)
  {
   for(ulong i=0; i<v.Size(); i++)
     v[i]=double(MathRand())/32767.;
  }
```

© 2000-2025, MetaQuotes Ltd.

Operaciones matemáticas

Las operaciones matemáticas (suma, resta, multiplicación y división) se pueden realizar en matrices y vectores miembro a miembro.

Ambas matrices o ambos vectores deberán ser del mismo tipo y tener las mismas dimensiones. Cada miembro de la matriz operará sobre el miembro correspondiente de la segunda matriz.

También podemos usar un escalar del tipo correspondiente (double, float o complex) como segundo término (multiplicador, sustraendo, divisor). En este caso, cada miembro de la matriz o vector operará en el escalar especificado.

```
   matrix matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
   matrix matrix_b={{1,2,3},{4,5,6}};

   matrix matrix_c1=matrix_a+matrix_b;
   matrix matrix_c2=matrix_b-matrix_a;
   matrix matrix_c3=matrix_a*matrix_b;    // producto de Hadamard (Hadamard product), no
   matrix matrix_c4=matrix_b/matrix_a;

   matrix_c1=matrix_a+1;
   matrix_c2=matrix_b-double_value;
   matrix_c3=matrix_a*M_PI;
   matrix_c4=matrix_b/0.1;

 //--- son posibles operaciones según el lugar
   matrix_a+=matrix_b;
   matrix_a/=2;
```

Las mismas operaciones están también disponibles para los vectores.

© 2000-2025, MetaQuotes Ltd.

Funciones matemáticas

Toda una serie de funciones matemáticas: MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, MathTanh pueden aplicarse a las matrices y vectores. En este caso, la matriz o vector se procesa por miembros.

En el caso de MathMod y MathPow, como segundo parámetro, podremos usar tanto un escalar como una matriz o vector del tamaño correspondiente

```
   matrix<T> mat1(128,128);
   matrix<T> mat3(mat1.Rows(),mat1.Cols());
   ulong     n,size=mat1.Rows()*mat1.Cols();
...
   mat2=MathPow(mat1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),(T)1.9);
      if(res!=mat2.Flat(n))
         errors++;
     }
   mat2=MathPow(mat1,mat3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),mat3.Flat(n));
      if(res!=mat2.Flat(n))
         errors++;
     }
...
   vector<T> vec1(16384);
   vector<T> vec3(vec1.Size());
   ulong     n,size=vec1.Size();
...
   vec2=MathPow(vec1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],(T)1.9);
      if(res!=vec2[n])
         errors++;
     }
   vec2=MathPow(vec1,vec3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],vec3[n]);
      if(res!=vec2[n])
         errors++;
     }
```

© 2000-2025, MetaQuotes Ltd.

Productos de matrices y vectores

Los cálculos de matrices y vectores incluyen:

- multiplicación de matrices
- multiplicación de vectores
- obtención de una matriz de covarianza
- cálculo de la correlación cruzada de dos vectores
- cálculo de la convolución de dos vectores
- cálculo del coeficiente de correlación

Función	Acción
MatMul	El método MatMul para realizar el producto de matrices y vectores tiene varias sobrecargas
GeMM	Producto matricial total de dos matrices (General Matrix Multiply)
Power	Eleva una matriz cuadrada a la potencia entera indicada
Dot	Producto escalar de dos vectores
Kron	Retorna el producto de Kronecker de dos matrices, una matriz y un vector, un vector y una matriz o dos vectores
Inner	Producto interior de dos matrices
Outer	Calcula el producto exterior de dos matrices o dos vectores
CorrCoef	Calcula el coeficiente de correlación de Pearson (coeficiente de correlación lineal)
Cov	Calcula la matriz de covarianza
Correlate	Calcula la función de correlación mutua (correlación cruzada) de dos vectores
Convolve	Retorna la convolución lineal discreta de dos vectores de secuencias

© 2000-2025, MetaQuotes Ltd.

MatMul

El método MatMul para realizar el producto de matrices y vectores tiene varias sobrecargas.

Multiplicación de una matriz por una matriz: `matrix[M][K] * matrix[K][N] = matrix[M][N]`

```
matrix  matrix::MatMul(
  const matrix&  b      // segunda matriz
   );
```

Multiplicación de un vector por una matriz: `horizontal vector[K] * matrix[K][N] = horizontal vector[N]`

```
vector  vector::MatMul(
  const matrix&  b      // matriz
   );
```

Multiplicación de una matriz por un vector: `matrix[M][K] * vertical vector[K] = vertical vector[M]`

```
vector  matrix::MatMul(
  const vector&  b      // vector
   );
```

Multiplicación escalar de vectores: `horizontal vector * vertical vector = dot value`

```
scalar  vector::MatMul(
  const vector&  b      // segundo vector
   );
```

Parámetros

b

[in] Matriz o vector.

Valor retornado

Matriz, vector o escalar, dependiendo del método utilizado.

Observación

Las matrices a multiplicar deberán ser compatibles, el número de columnas de la primera matriz deberá ser igual al número de filas de la segunda matriz. La multiplicación de matrices es no conmutativa: la multiplicación de la primera matriz por la segunda matriz no es en general igual a la multiplicación de la segunda matriz por la primera.

El producto matricial consiste en todas las combinaciones posibles de los productos escalares de los vectores de fila de la primera matriz y los vectores de columna de la segunda matriz.

En la multiplicación escalar de vectores, sus longitudes deberán coincidir.

Al multiplicar un vector y una matriz, la longitud del vector deberá coincidir exactamente con el número de columnas de la matriz.

© 2000-2025, MetaQuotes Ltd.

Algoritmo ingenuo de multiplicación de matrices en MQL5:

```
matrix MatrixProduct(const matrix& matrix_a, const matrix& matrix_b)
  {
   matrix matrix_c;

   if(matrix_a.Cols()!=matrix_b.Rows())
      return(matrix_c);

   ulong M=matrix_a.Rows();
   ulong K=matrix_a.Cols();
   ulong N=matrix_b.Cols();
   matrix_c=matrix::Zeros(M,N);

   for(ulong m=0; m<M; m++)
      for(ulong k=0; k<K; k++)
         for(ulong n=0; n<N; n++)
            matrix_c[m][n]+=matrix_a[m][k]*matrix_b[k][n];

   return(matrix_c);
  }
```

Ejemplo de multiplicación de matrices

```
   matrix a={{1, 0, 0},
             {0, 1, 0}};
   matrix b={{4, 1},
             {2, 2},
             {1, 3}};
   matrix c1=a.MatMul(b);
   matrix c2=b.MatMul(a);
   Print("c1 = \n", c1);
   Print("c2 = \n", c2);
/*
   c1 =
   [[4,1]
    [2,2]]
   c2 =
   [[4,1,0]
    [2,2,0]
    [1,3,0]]
*/
```

Ejemplo de multiplicación de un vector horizontal por una matriz

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- creamos una matriz de 3x5
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   vector v3 = {1, 2, 3};
   Print("Producto del vector horizontal v por la matriz m[3,5]");
   Print("A la izquierda vector v3 = ", v3);
   Print("A la derecha matriz m35 = \n", m35);
   Print("v3.MatMul(m35) = vector horizontal v[5] \n", v3.MatMul(m35));

  /* Resultado
    Producto del vector horizontal v3 por la matriz m[3,5]
    A la izquierda vector v3 = [1,2,3]
    A la derecha matriz m35 =
    [[0,1,2,3,4]
     [5,6,7,8,9]
     [10,11,12,13,14]]
    v3.MatMul(m35) = vector horizontal v[5]
    [40,46,52,58,64]
   */
  }
//+------------------------------------------------------------------+
//|  Rellenamos la matriz con valores incrementales                  |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

Ejemplo de multiplicación de una matriz por un vector vertical

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- creamos una matriz de 3x5
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   Print("Producto de la matriz m[3,5] por el vector vertical v[5]");
   vector v5 = {1,2,3,4,5};
   Print("A la izquierda m35 = \n",m35);
   Print("A la derecha v5 = ",v5);
   Print("m35.MatMul(v5) = vector vertical v[3] \n",m35.MatMul(v5));

  /* Resultado
   Producto de la matriz m[3,5] por el vector vertical v[5]
   A la izquierda m35 =
   [[0,1,2,3,4]
    [5,6,7,8,9]
    [10,11,12,13,14]]
   A la derecha v5 = [1,2,3,4,5]
   m35.MatMul(v5) = vector vertical v[3]
   [40,115,190]
   */
  }
//+------------------------------------------------------------------+
//|  Rellenamos la matriz con valores incrementales                  |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

Ejemplo de multiplicación escalar de vectores

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- multiplicación escalar de vectores horizontales y verticales
   vector a= {1, 2, 3};  // vector horizontal
   vector b= {4, 5, 6};  // vector vertical
   Print("a = ", a);
   Print("b = ", b);
   Print("1) a.MatMul(b) = ", a.MatMul(b));
   //--- mostramos que el método Dot da el mismo resultado
   Print("2) a.Dot(b) = ", a.Dot(b));

  /* Resultado
   a = [1,2,3]
   b = [4,5,6]
   1) a.MatMul(b) = 32.0
   2) a.Dot(b) = 32.0
    */
  }
```

Ver también

Dot, GeMM

© 2000-2025, MetaQuotes Ltd.

GeMM

El método GeMM realiza el producto matricial total de dos matrices (General Matrix Multiply). En términos generales, la expresión se escribe como C ← α A B + ß C , donde las matrices A y B pueden transponerse opcionalmente. Cuando las matrices AB se multiplican de forma normal (MatMul), se presupone que el coeficiente *alpha* es igual a la unidad, mientras que *beta* es igual a cero.

La principal diferencia entre GeMM y MatMul en términos de eficiencia es que MatMul siempre crea un nuevo objeto matriz/vector, mientras que GeMM trabaja con un objeto de matriz existente que no se vuelve a crear. Por consiguiente, al utilizarse con GeMM, si asignamos memoria para la matriz correspondiente por adelantado y luego trabajamos con los mismos tamaños de matriz, no habrá reasignación de memoria. Esto puede ser una ventaja muy importante a favor de GeMM en los cálculos masivos, por ejemplo, al optimizar en el simulador de estrategias o entrenar redes neuronales.

GeMM también tiene 4 sobrecargas como el método MatMul. No obstante, la semántica de la 4ª sobrecarga se ha modificado para permitir multiplicar vectores verticales y horizontales.

En un objeto de matriz/vector existente, no será necesario asignar memoria para los datos por adelantado. La primera vez que se llame a GeMM, la memoria será asignada y rellenada con ceros.

Multiplicación de una matriz por una matriz: `matrix C[M][N] = α * ( matrix A[M][K] * matrix B[K][N]) + ß * matrix C[M][N]`

```
bool  matrix::GeMM(
  const matrix &A,     // primera matriz
  const matrix &B,     // segunda  matriz
  double alpha,        // multiplicador alfa del producto de matrices AB
  double beta,         // multiplicador beta de la matriz C
  uint   flags         // combinación de valores de la enumeración ENUM_GEMM (de bits '
   );
```

Multiplicación de un vector por una matriz: `vector C[N] = α * ( vector A[K] * matrix B[K][N]) + ß * vector C[N]`

```
bool  vector::GeMM(
  const vector &A,     // vector horizontal
  const matrix &B,     // matriz
  double alpha,        // multiplicador alfa del producto AB
  double beta,         // multiplicador beta para el vector C
  uint   flags         // valor de la enumeración ENUM_GEMM  que determina la transpon
   );
```

Multuplicación de una matriz por un vector: `vector C[M] = α * ( matrix A[M][K] * vector B[K] * ) + ß * vector C[M]`

```
bool  vector::GeMM(
  const matrix &A,     // matriz
  const vector &B,     // vector vertical
  double alpha,        // multiplicador alfa del producto AB
  double beta,         // multiplicador beta para el vector C
  uint   flags         // valor de la enumeración ENUM_GEMM  que determina la transpon
   );
```

© 2000-2025, MetaQuotes Ltd.

Multiplicación de un vector por un vector: `matrix C[M][N] = α * ( vector A[M] * vector B[N] * ) + β * matrix C[M][N]`. Esta sobrecarga retorna una matriz, a diferencia del método MatMul, que retorna un escalar.

```
bool  matrix::GeMM(
  const vector &A,     // primer vector
  const vector &B,     // segundo  vector
  double alpha,        // multiplicador alfa del producto AB
  double beta,         // multiplicador beta de la matriz C
  uint   flags         // valor de la enumeración ENUM_GEMM  que determina la transponi
   );
```

Parámetros

A

[in] Matriz o vector.

B

[in] Matriz o vector.

alpha

[in] Multiplicador alfa para el producto AB.

beta

[in] Multiplicador beta para la matriz resultante C.

flags

[in] Valor de la enumeración ENUM_GEMM que determina la transponibilidad de las matrices A, B y C.

Valor retornado

true si la operación se ha ejecutado con éxito, de lo contrario, false.

ENUM_GEMM

Enumeración de banderas para el método GeMM.

Identificador	Descripción
TRANSP_A	Utilizar la matriz transpuesta A
TRANSP_B	Utilizar la matriz transpuesta B
TRANSP_C	Utilizar la matriz transpuesta C

Observación

Como parámetros A y B podemos utilizar matrices y vectores de tipo float, double y complex. Así, las variantes de plantilla del método GeMM serán las siguientes:

© 2000-2025, MetaQuotes Ltd.

```
bool matrix<T>::GeMM(const matrix<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags)
bool matrix<T>::GeMM(const vector<T> &A,const vector<T> &B,T alpha,T beta,ulong flags)

bool vector<T>::GeMM(const vector<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags)
bool vector<T>::GeMM(const matrix<T> &A,const vector<T> &B,T alpha,T beta,ulong flags)
```

En general, una función general de multiplicación de matrices se describirá como:

```
C[m,n] = α *Sum(A[m,k]*B[k,n]) + β*C[m,n]
```

donde la matriz A es de tamaño M x K, la matriz B es K x N, y la matriz C es M x N.

Así, las matrices a multiplicar deberán ser compatibles, el número de columnas de la primera matriz deberá ser igual al número de filas de la segunda matriz. La multiplicación de matrices es no conmutativa: la multiplicación de la primera matriz por la segunda matriz no es en general igual a la multiplicación de la segunda matriz por la primera.

Ejemplo:

```
void OnStart()
  {
   vector vector_a= {1, 2, 3, 4, 5};
   vector vector_b= {4, 3, 2, 1};
   matrix matrix_c;
//--- calculamos GeMM para los dos vectores
   matrix_c.GeMM(vector_a, vector_b, 1, 0);
   Print("matrix_c:\n ", matrix_c, "\n");
   /*
   matrix_c:
    [[4,3,2,1]
    [8,6,4,2]
    [12,9,6,3]
    [16,12,8,4]
    [20,15,10,5]]
   */
//--- creamos las matrices en forma de vectores
   matrix matrix_a(5, 1);
   matrix matrix_b(1, 4);
   matrix_a.Col(vector_a, 0);
   matrix_b.Row(vector_b, 0);
   Print("matrix_a:\n ", matrix_a);
   Print("matrix_b:\n ", matrix_b);
   /*
   matrix_a:
   [[1]
   [2]
   [3]
```

© 2000-2025, MetaQuotes Ltd.

```
    [4]
    [5]]
    matrix_b:
    [[4,3,2,1]]
    */
//-- calculamos GeMM para las dos matrices y obtemos el mismo resultado
    matrix_c.GeMM(matrix_a, matrix_b, 1, 0);
    Print("matrix_c:\n ", matrix_c);
    /*
    matrix_c:
     [[4,3,2,1]
     [8,6,4,2]
     [12,9,6,3]
     [16,12,8,4]
     [20,15,10,5]]
    */
  }
```

Ver también

MatMul

© 2000-2025, MetaQuotes Ltd.

Power

Eleva una matriz cuadrada a la potencia entera indicada.

```
matrix matrix::Power(
  const int  power      // potencia
   );
```

Parámetros

power

[in] El exponente puede ser cualquier número entero: positivo, negativo o cero.

Valor retornado

Matriz.

Observación

La matriz resultante tendrá el mismo tamaño que la matriz original. En caso de elevar la matriz a la potencia 0, se retornará una matriz unitaria. La potencia positiva n significa que la matriz original se multiplica n veces por sí misma. La potencia negativa -n significa que primero se invierte la matriz original y luego la matriz invertida se multiplica por sí misma n veces.

Aquí vemos un algoritmo simple para la multiplicación de matrices en MQL5:

```
bool MatrixPower(matrix& c, const matrix& a, const int power)
  {
//--- la matriz deberá ser cuadrada
   if(a.Rows()!=a.Cols())
      return(false);
//--- el tamaño de la matriz resultante es exactamente el mismo
   ulong  rows=a.Rows();
   ulong  cols=a.Cols();
   matrix result(rows,cols);
//--- con una potencia cero, retornaremos una matriz unitaria
   if(power==0)
      result.Identity();
   else
     {
      //--- con una potencia cero, primero invertiremos la matriz
      if(power<0)
        {
         matrix inverted=a.Inv();
         result=inverted;
         for(int i=-1; i>power; i--)
            result=result.MatMul(inverted);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      else
        {
         result=a;
         for(int i=1; i<power; i++)
            result=result.MatMul(a);
        }
     }
//---
   c=result;
   return(true);
  }
```

Ejemplo en MQL5:

```
  matrix i= {{0, 1}, {-1, 0}};
  Print("i:\n", i);

  Print("i.Power(3):\n", i.Power(3));

  Print("i.Power(0):\n", i.Power(0));

  Print("i.Power(-3):\n", i.Power(-3));

  /*
  i:
  [[0,1]
   [-1,0]]

  i.Power(3):
  [[0,-1]
   [1,0]]

  i.Power(0):
  [[1,0]
   [0,1]]

  i.Power(-3):
  [[0, -1]
   [1,0]]
  */
```

Ejemplo en Python:

```
import numpy as np
from numpy.linalg import matrix_power

# matrix equiv. of the imaginary unit
```

© 2000-2025, MetaQuotes Ltd.

```
i = np.array([[0, 1], [-1, 0]])
print("i:\n",i)

# should = -i
print("matrix_power(i, 3) :\n",matrix_power(i, 3) )

print("matrix_power(i, 0):\n",matrix_power(i, 0))

# should = 1/(-i) = i, but w/ f.p. elements
print("matrix_power(i, -3):\n",matrix_power(i, -3))

i:
 [[ 0  1]
 [-1  0]]

matrix_power(i, 3) :
 [[ 0 -1]
 [ 1  0]]

matrix_power(i, 0):
 [[1 0]
 [0 1]]

matrix_power(i, -3):
 [[ 0.  1.]
 [-1.  0.]]
```

© 2000-2025, MetaQuotes Ltd.

Dot

Producto escalar de dos vectores.

```
double vector::Dot(
  const vector&  b      // segundo vector
   );
```

Parámetros

b

[in] Vector.

Valor retornado

Escalar.

Observación

El producto dot para dos matrices no es otra cosa que el producto matricial matrix::MatMul().

Aquí tenemos un algoritmo sencillo para el producto escalar de vectores en MQL5:

```
double VectorDot(const vector& vector_a, const vector& vector_b)
  {
   double dot=0;

   if(vector_a.Size()==vector_b.Size())
     {
      for(ulong i=0; i<vector_a.Size(); i++)
         dot+=vector_a[i]*vector_b[i];
     }

   return(dot);
  }
```

Ejemplo en MQL5:

```
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

Ejemplo en Python:

```
import numpy as np

a = [1, 0, 0, 1]
b = [4, 1, 2, 2]
print(np.dot(a, b))

>>> 6
```

© 2000-2025, MetaQuotes Ltd.

Kron

Retorna el producto de Kronecker de dos matrices, una matriz y un vector, un vector y una matriz o dos vectores.

```
matrix matrix::Kron(
  const matrix&  b      // segunda matriz
   );

matrix matrix::Kron(
  const vector&  b      // vector
   );

matrix vector::Kron(
  const matrix&  b      // matriz
   );

matrix vector::Kron(
  const vector&  b      // segundo vector
   );
```

Parámetros

b

[in] Segunda matriz.

Valor retornado

Matriz.

Observación

El producto de Kronecker también se denomina multiplicación de matrices en bloque.

Aquí vemos un sencillo algoritmo del producto de Kronecker para dos matrices en MQL5:

```
matrix MatrixKronecker(const matrix& matrix_a,const matrix& matrix_b)
  {
   ulong  M=matrix_a.Rows();
   ulong  N=matrix_a.Cols();
   ulong  P=matrix_b.Rows();
   ulong  Q=matrix_b.Cols();
   matrix matrix_c(M*P,N*Q);

   for(ulong m=0; m<M; m++)
      for(ulong n=0; n<N; n++)
         for(ulong p=0; p<P; p++)
```

© 2000-2025, MetaQuotes Ltd.

```
            for(ulong q=0; q<Q; q++)
               matrix_c[m*P+p][n*Q+q]=matrix_a[m][n] * matrix_b[p][q];

   return(matrix_c);
  }
```

Ejemplo en MQL5:

```
   matrix a={{1,2,3},{4,5,6}};
   matrix b=matrix::Identity(2,2);
   vector v={1,2};

   Print(a.Kron(b));
   Print(a.Kron(v));

  /*
   [[1,0,2,0,3,0]
    [0,1,0,2,0,3]
    [4,0,5,0,6,0]
    [0,4,0,5,0,6]]

   [[1,2,2,4,3,6]
    [4,8,5,10,6,12]]
  */
```

Ejemplo en Python:

```
import numpy as np

A = np.arange(1,7).reshape(2,3)
B = np.identity(2)
V = [1,2]
print(np.kron(A, B))
print("")
print(np.kron(A, V))

[[1. 0. 2. 0. 3. 0.]
 [0. 1. 0. 2. 0. 3.]
 [4. 0. 5. 0. 6. 0.]
 [0. 4. 0. 5. 0. 6.]]

[[ 1  2  2  4  3  6]
 [ 4  8  5 10  6 12]]
```

© 2000-2025, MetaQuotes Ltd.

Inner

Producto interior de dos matrices.

```
matrix matrix::Inner(
  const matrix&  b      // segunda matriz
   );
```

Parámetros

b

[in] Matriz.

Valor retornado

Matriz.

Observación

El producto interior de dos vectores no es más que el producto escalar de dos vectores vector::Dot().

Aquí tenemos un algoritmo simple para el producto interior de dos matrices en MQL5:

```
bool MatrixInner(matrix& c, const matrix& a, const matrix& b)
  {
//--- el número de columnas deberá ser el mismo
   if(a.Cols()!=b.Cols())
      return(false);
//--- el tamaño de la matriz resultante dependerá del número de vectores de cada matri
   ulong  rows=a.Rows();
   ulong  cols=b.Rows();
   matrix result(rows,cols);
//---
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
//---
   c=result;
   return(true);
  }
```

Ejemplo en MQL5:

© 2000-2025, MetaQuotes Ltd.

```
   matrix a={{0,1,2},{3,4,5}};
   matrix b={{0,1,2},{3,4,5},{6,7,8}};
   matrix c=a.Inner(b);
   Print(c);
   matrix a1={{0,1,2}};
   matrix c1=a1.Inner(b);
   Print(c1);

  /*
  [[5,14,23]
  [14,50,86]]
  [[5,14,23]]
  */
```

Ejemplo en Python:

```
import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))

import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))
```

© 2000-2025, MetaQuotes Ltd.

Outer

Calcula el producto exterior de dos matrices o dos vectores.

```
matrix matrix::Outer(
  const matrix&  b      // segunda matriz
   );

matrix vector::Outer(
  const vector&  b      // segundo vector
   );
```

Parámetros

b

[in] Matriz.

Valor retornado

Matriz.

Observación

El producto externo, como el producto de Kronecker, es también una multiplicación en bloque de matrices (y vectores).

Aquí tenemos un algoritmo de producto externo de dos matrices en MQL5:

```
matrix MatrixOuter(const matrix& matrix_a, const matrix& matrix_b)
  {
//--- el tamaño de la matriz resultante dependerá del tamaño de las matrices
   ulong  rows=matrix_a.Rows()*matrix_a.Cols();
   ulong  cols=matrix_b.Rows()*matrix_b.Cols();
   matrix matrix_c(rows,cols);
   ulong  cols_a=matrix_a.Cols();
   ulong  cols_b=matrix_b.Cols();
//---
   for(ulong i=0; i<rows; i++)
     {
      ulong row_a=i/cols_a;
      ulong col_a=i%cols_a;
      for(ulong j=0; j<cols; j++)
        {
         ulong row_b=j/cols_b;
         ulong col_b=j%cols_b;
         matrix_c[i][j]=matrix_a[row_a][col_a] * matrix_b[row_b][col_b];
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//---
   return(matrix_c);
  }
```

Ejemplo en MQL5:

```
   vector vector_a={0,1,2,3,4,5};
   vector vector_b={0,1,2,3,4,5,6};
   Print("vector_a.Outer\n",vector_a.Outer(vector_b));
   Print("vector_a.Kron\n",vector_a.Kron(vector_b));

   matrix matrix_a={{0,1,2},{3,4,5}};
   matrix matrix_b={{0,1,2},{3,4,5},{6,7,8}};
   Print("matrix_a.Outer\n",matrix_a.Outer(matrix_b));
   Print("matrix_a.Kron\n",matrix_a.Kron(matrix_b));

  /*
   vector_a.Outer
   [[0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6]
    [0,2,4,6,8,10,12]
    [0,3,6,9,12,15,18]
    [0,4,8,12,16,20,24]
    [0,5,10,15,20,25,30]]
   vector_a.Kron
   [[0,0,0,0,0,0,0,0,1,2,3,4,5,6,0,2,4,6,8,10,12,0,3,6,9,12,15,18,0,4,8,12,16,20,24,0,
   matrix_a.Outer
   [[0,0,0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6,7,8]
    [0,2,4,6,8,10,12,14,16]
    [0,3,6,9,12,15,18,21,24]
    [0,4,8,12,16,20,24,28,32]
    [0,5,10,15,20,25,30,35,40]]
   matrix_a.Kron
   [[0,0,0,0,1,2,0,2,4]
    [0,0,0,3,4,5,6,8,10]
    [0,0,0,6,7,8,12,14,16]
    [0,3,6,0,4,8,0,5,10]
    [9,12,15,12,16,20,15,20,25]
    [18,21,24,24,28,32,30,35,40]]
   */
```

Ejemplo en Python:

```
import numpy as np
```

© 2000-2025, MetaQuotes Ltd.

```
A = np.arange(6)
B = np.arange(7)
print("np.outer")
print(np.outer(A, B))
print("np.kron")
print(np.kron(A, B))

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
print("np.outer")
print(np.outer(A, B))
print("np.kron")

np.outer
[[ 0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6]
 [ 0  2  4  6  8 10 12]
 [ 0  3  6  9 12 15 18]
 [ 0  4  8 12 16 20 24]
 [ 0  5 10 15 20 25 30]]
np.kron
[ 0  0  0  0  0  0  0  0  1  2  3  4  5  6  0  2  4  6  8 10 12  0  3  6
  9 12 15 18  0  4  8 12 16 20 24  0  5 10 15 20 25 30]
np.outer
[[ 0  0  0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6  7  8]
 [ 0  2  4  6  8 10 12 14 16]
 [ 0  3  6  9 12 15 18 21 24]
 [ 0  4  8 12 16 20 24 28 32]
 [ 0  5 10 15 20 25 30 35 40]]
np.kron
[[ 0  0  0  0  1  2  0  2  4]
 [ 0  0  0  3  4  5  6  8 10]
 [ 0  0  0  6  7  8 12 14 16]
 [ 0  3  6  0  4  8  0  5 10]
 [ 9 12 15 12 16 20 15 20 25]
 [18 21 24 24 28 32 30 35 40]]
```

© 2000-2025, MetaQuotes Ltd.

CorrCoef

Calcula el coeficiente de correlación de Pearson (coeficiente de correlación lineal).

```
matrix matrix::CorrCoef(
  const bool    rowvar=true  // los vectores de observaciones están ubicados en la fi
);

scalar vector::CorrCoef(
  const vector&  b           // segundo vector
);
```

Parámetros

Parámetros

rowvar

[in] Bandera que define la posición de los vectores en la matriz para calcular la correlación por pares, horizontal o verticalmente. Por defecto *rowvar* es true, esto significa que los vectores estarán en las filas de la matriz. Si *rowvar* es false, las correlaciones se contarán entre las columnas de la matriz.

b

[in] Segundo vector.

Valor retornado

Valor de los coeficientes de correlación para un vector.

Matriz cuadrada de los valores de los coeficientes de correlación entre los vectores para una matriz. El tamaño de la matriz se corresponde con el número de vectores de la matriz que pueden disponerse en las filas (*rowvar*=true) o en las columnas ((*rowvar*=false).

Observación

El coeficiente de correlación estará en el rango [-1, 1].

Debido al redondeo del número con coma flotante, la matriz resultante podría no ser hermitiana, los elementos diagonales podrían no ser iguales a 1, y los elementos podrían no cumplir la desigualdad abs(a) <= 1. Las partes real e imaginaria se truncan hasta el intervalo [-1, 1] para solucionar esto, pero no siempre resulta de ayuda en los casos complejos.

Aquí vemos un algoritmo simple para calcular el coeficiente de correlación de dos vectores en MQL5:

```
double VectorCorrelation(const vector& vector_x,const vector& vector_y)
  {
   ulong n=vector_x.Size()<vector_y.Size() ? vector_x.Size() : vector_y.Size();
   if(n<=1)
```

© 2000-2025, MetaQuotes Ltd.

```
      return(0);

   ulong  i;
   double xmean=0;
   double ymean=0;
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_x[i]))
         return(0);
      if(!MathIsValidNumber(vector_y[i]))
         return(0);
      xmean+=vector_x[i];
      ymean+=vector_y[i];
     }
   xmean/=(double)n;
   ymean/=(double)n;

   double s=0;
   double xv=0;
   double yv=0;
   double t1=0;
   double t2=0;
//--- cálculo
   s=0;
   for(i=0; i<n; i++)
     {
      t1=vector_x[i]-xmean;
      t2=vector_y[i]-ymean;
      xv+=t1*t1;
      yv+=t2*t2;
      s+=t1*t2;
     }
//--- comprobación
   if(xv==0 || yv==0)
      return(0);
//--- retornamos el resultado
   return(s/(MathSqrt(xv)*MathSqrt(yv)));
  }
```

Ejemplo en MQL5:

```
   vectorf vector_a={1,2,3,4,5};
   vectorf vector_b={0,1,0.5,2,2.5};
   Print("vectors correlation ",vector_a.CorrCoef(vector_b));
//---
   matrixf matrix_a={{1,2,3,4,5},
                     {0,1,0.5,2,2.5}};
   Print("matrix rows correlation\n",matrix_a.CorrCoef());
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixf matrix_a2=matrix_a.Transpose();
   Print("transposed matrix cols correlation\n",matrix_a2.CorrCoef(false));
   matrixf matrix_a3={{1.0f, 2.0f, 3.0f, 4.0f, 5.0f},
                      {0.0f, 1.0f, 0.5f, 2.0f, 2.5f},
                      {0.1f, 1.0f, 2.0f, 1.0f, 0.3f}};
   Print("rows correlation\n",matrix_a3.CorrCoef());
   Print("cols correlation\n",matrix_a3.CorrCoef(false));

  /*
   vectors correlation 0.9149913787841797
   matrix rows correlation
   [[1,0.91499138]
    [0.91499138,1]]
   transposed matrix cols correlation
   [[1,0.91499138]
    [0.91499138,1]]
   rows correlation
   [[1,0.91499138,0.08474271]
    [0.91499138,1,-0.17123166]
    [0.08474271,-0.17123166,1]]
   cols correlation
   [[1,0.99587059,0.85375023,0.91129309,0.83773589]
    [0.99587059,1,0.80295509,0.94491106,0.88385159]
    [0.85375023,0.80295509,1,0.56362146,0.43088508]
    [0.91129309,0.94491106,0.56362146,1,0.98827404]
    [0.83773589,0.88385159,0.43088508,0.98827404,1]]
   */
```

Ejemplo en Python:

```
import numpy as np
va=[1,2,3,4,5]
vb=[0,1,0.5,2,2.5]
print("vectors correlation")
print(np.corrcoef(va,vb))

ma=np.zeros((2,5))
ma[0,:]=va
ma[1,:]=vb
print("matrix rows correlation")
print(np.corrcoef(ma))
print("transposed matrix cols correlation")
print(np.corrcoef(np.transpose(ma),rowvar=False))
print("")

ma1=[[1,2,3,4,5],[0,1,0.5,2,2.5],[0.1,1,0.2,1,0.3]]
print("rows correlation\n",np.corrcoef(ma1))
print("cols correlation\n",np.corrcoef(ma1,rowvar=False))
```

© 2000-2025, MetaQuotes Ltd.

```
transposed matrix cols correlation
[[1.         0.91499142]
 [0.91499142 1.        ]]

rows correlation
 [[1.         0.91499142 0.1424941 ]
 [0.91499142 1.         0.39657517]
 [0.1424941  0.39657517 1.        ]]
cols correlation
 [[1.         0.99587059 0.98226063 0.91129318 0.83773586]
 [0.99587059 1.         0.99522839 0.94491118 0.88385151]
 [0.98226063 0.99522839 1.         0.97234063 0.92527551]
 [0.91129318 0.94491118 0.97234063 1.         0.98827406]
 [0.83773586 0.88385151 0.92527551 0.98827406 1.        ]]
```

© 2000-2025, MetaQuotes Ltd.

Cov

Calcula la matriz de covarianza.

```
matrix matrix::Cov(
  const bool    rowvar=true  // filas o columnas de los vectores de observación
);

matrix matrix::Cov(
  const bool    rowvar,      // filas o columnas de los vectores de observación
  const int     ddof         // desplazamiento del grado de libertad
);

matrix vector::Cov(
  const vector&  b           // segundo vector
);

matrix vector::Cov(
  const vector&  b,           // segundo vector
  const int      ddof         // desplazamiento del grado de libertad
);
```

Parámetros

rowwar

[in] Si el valor *rowwar* es igual a true (por defecto), entonces cada línea será una variable y las observaciones se encontrarán a lo largo de las columnas. De lo contrario, (si es igual a false), las líneas contendrán observaciones, y las columnas serán variables (es decir, la matriz se transpondrá).

b

[in] Segundo vector.

ddof

[in] "Delta Degrees of Freedom": parámetro que determina cuánto disminuye el número de grados de libertad. El divisor en la fórmula será N - ddof, donde N será el número de observaciones. Por defecto DDOF = 1.

Observación

Calcula la matriz de covarianza.

Aquí tenemos un algoritmo sencillo para calcular la matriz de covarianza de dos vectores en MQL5:

```
bool VectorCovariation(const vector& vector_a,const vector& vector_b,matrix& matrix_c)
  {
   int i,j;
   int m=2;
   int n=(int)(vector_a.Size()<vector_b.Size()?vector_a.Size():vector_b.Size());
```

© 2000-2025, MetaQuotes Ltd.

```
//--- comprobaciones
   if(n<=1)
      return(false);
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_a[i]))
         return(false);
      if(!MathIsValidNumber(vector_b[i]))
         return(false);
     }
//---
   matrix matrix_x(2,n);
   matrix_x.Row(vector_a,0);
   matrix_x.Row(vector_b,1);
   vector t=vector::Zeros(m);
//--- cálculo
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
         t[i]+=matrix_x[i][j]/double(n);
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
         matrix_x[i][j]-=t[i];
//--- syrk C=alpha*A^H*A+beta*C (beta=0 and not considered)
   matrix_c=matrix::Zeros(m,m);
   for(i=0; i<m; i++)
     {
      for(j=0; j<n; j++)
        {
         double v=matrix_x[i][j]/(n-1);
         for(int i_=i; i_<m; i_++)
            matrix_c[i][i_]+=v*matrix_x[i_][j];
        }
     }
//--- force symmetricity
   for(i=0; i<m-1; i++)
      for(j=i+1; j<m; j++)
         matrix_c[j][i]=matrix_c[i][j];
//---
   return(true);
  }
```

Ejemplo en MQL5:

```
   matrix matrix_a={{3,-2.1},{1.1,-1},{0.12,4.3}};
   Print("covariation cols\n",matrix_a.Cov(false));
   Print("covariation rows\n",matrix_a.Cov());

   vector vector_a=matrix_a.Col(0);
```

© 2000-2025, MetaQuotes Ltd.

```
   vector vector_b=matrix_a.Col(1);
   Print("covariation vectors\n",vector_a.Cov(vector_b));

  /*
   covariation cols
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
   covariation rows
   [[13.005,5.355,-10.659]
    [5.355,2.205,-4.389]
    [-10.659,-4.389,8.736199999999998]]
   covariation vectors
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
  */
```

Ejemplo en Python:

```
import numpy as np
matrix_a=np.array([[3,-2.1],[1.1,-1],[0.12,4.3]])
matrix_c=np.cov(matrix_a,rowvar=False)
print("covariation cols\n",matrix_c)
matrix_c2=np.cov(matrix_a)
print("covariation rows\n",matrix_c2)

vector_a=matrix_a[:,0]
vector_b=matrix_a[:,1]
matrix_c3=np.cov(vector_a,vector_b)
print("covariation vectors\n",matrix_c3)

covariation cols
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
covariation rows
 [[ 13.005    5.355  -10.659 ]
 [  5.355    2.205   -4.389 ]
 [-10.659   -4.389    8.7362]]
covariation vectors
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
```

© 2000-2025, MetaQuotes Ltd.

Correlate

Calcula la función de correlación mutua (correlación cruzada) de dos vectores.

```
vector vector::Correlate(
  const vector&           v,        // vector
  ENUM_VECTOR_CONVOLVE    mode      // modo
   );
```

Parámetros

v

[in] Segundo vector.

mode

[in] El parámetro "mode" define el modo de cálculo de la convolución lineal. Valor de la enumeración ENUM_VECTOR_CONVOLVE.

Valor retornado

Correlación cruzada de dos vectores.

Observación

El parámetro "mode" define el modo de cálculo de la convolución lineal.

Aquí tenemos un algoritmo sencillo para calcular la correlación cruzada de dos vectores en MQL5:

```
vector VectorCrossCorrelationFull(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[n-i-1]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationSame(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n);
```

© 2000-2025, MetaQuotes Ltd.

```
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationValid(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
```

Ejemplo en MQL5:

```
   vector a={1,2,3,4,5};
   vector b={0,1,0.5};

   Print("full\n",a.Correlate(b,VECTOR_CONVOLVE_FULL));
   Print("same\n",a.Correlate(b,VECTOR_CONVOLVE_SAME));
   Print("valid\n",a.Correlate(b,VECTOR_CONVOLVE_VALID));
   Print("full\n",b.Correlate(a,VECTOR_CONVOLVE_FULL));

  /*
```

© 2000-2025, MetaQuotes Ltd.

```
   full
   [0.5,2,3.5,5,6.5,5,0]
   same
   [2,3.5,5,6.5,5]
   valid
   [3.5,5,6.5]
   full
   [0,5,6.5,5,3.5,2,0.5]
  */
```

Ejemplo en Python:

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.correlate(a,b,'full'))
print("same\n",np.correlate(a,b,'same'));
print("valid\n",np.correlate(a,b,'valid'));
print("full\n",np.correlate(b,a,'full'))

full
 [0.5 2.  3.5 5.  6.5 5.  0. ]
same
 [2.  3.5 5.  6.5 5. ]
valid
 [3.5 5.  6.5]
full
 [0.  5.  6.5 5.  3.5 2.  0.5]
```

© 2000-2025, MetaQuotes Ltd.

Convolve

Retorna la convolución lineal discreta de dos vectores de secuencias.

```
vector vector::Convolve(
  const vector&            v,         // vector
  ENUM_VECTOR_CONVOLVE     mode       // modo
   );
```

Parámetros

v

[out] Segundo vector.

mode

[in] El parámetro de modo determina el modo de cálculo de convolución lineal ENUM_VECTOR_CONVOLVE.

Valor retornado

Convolución lineal discreta de dos vectores.

Algoritmo simple para calcular la convolución de dos vectores en MQL5:

```
vector VectorConvolutionFull(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionFull(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[i]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionSame(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionSame(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
```

© 2000-2025, MetaQuotes Ltd.

```
   int    size=MathMax(m,n);
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionValid(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionValid(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
```

Ejemplo en MQL5:

```
  vector a= {1, 2, 3, 4, 5};
  vector b= {0, 1, 0.5};

  Print("full\n", a.Convolve(b, VECTOR_CONVOLVE_FULL));
  Print("same\n", a.Convolve(b, VECTOR_CONVOLVE_SAME));
```

© 2000-2025, MetaQuotes Ltd.

```
  Print("valid\n", a.Convolve(b, VECTOR_CONVOLVE_VALID));

  /*
   full
   [0,1,2.5,4,5.5,7,2.5]
   same
   [1,2.5,4,5.5,7]
   valid
   [2.5,4,5.5]
  */
```

Ejemplo en Python:

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.convolve(a,b,'full'))
print("same\n",np.convolve(a,b,'same'));
print("valid\n",np.convolve(a,b,'valid'));


full
 [0.  1.  2.5 4.  5.5 7.  2.5]
same
 [1.  2.5 4.  5.5 7. ]
valid
 [2.5 4.  5.5]
```

© 2000-2025, MetaQuotes Ltd.

Transformación de matrices

La descomposición de matrices es una tarea que puede plantearse en los siguientes casos:

- como paso intermedio en el proceso de resolución de sistemas de ecuaciones lineales
- para invertir matrices
- para calcular los determinantes
- al buscar los valores y vectores propios de las matrices
- para calcular funciones analíticas sobre matrices
- al utilizar el método de los mínimos cuadrados
- al solucionar numéricamente ecuaciones diferenciales

Además, dependiendo del problema a resolver, se usarán distintos tipos de descomposición de matrices.

Función	Acción
Cholesky	Calcula la descomposición de Cholesky
Eig	Calcula los valores propios y los vectores propios derechos de una matriz cuadrada
EigVals	Calcula los valores propios de la matriz global
LU	Realiza una factorización LU de una matriz como producto de una matriz rectangular inferior y una matriz rectangular superior.
LUP	Realiza una factorización LUP con rotación parcial, que es una factorización LU solo con permutaciones de líneas: PA=LU
QR	Calcula la factorización QR de una matriz
SVD	Calcula la descomposición de valores singulares

© 2000-2025, MetaQuotes Ltd.

Cholesky

Calcula la descomposición de Cholesky.

```
bool matrix::Cholesky(
   matrix&  L      // matrix
   );
```

Parámetros

L

[out] Matriz triangular inferior.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

Retorna la expansión de Cholesky, L * L.H, de una matriz cuadrada A, donde L es la matriz rectangular inferior y .H es la matriz conjugada hermitiana (que es la transposición habitual cuando A es real). A debe ser hermitiana (simétrica, si es real) y positivamente definida. La función no comprueba si A es hermitiana o no. Además, solo se utilizan los elementos rectangulares inferiores y diagonales de A. De hecho, solo se retorna L.

Ejemplo:

```
matrix matrix_a= {{5.7998084, -2.1825367}, {-2.1825367, 9.85910595}};
matrix matrix_l;
Print("matrix_a\n", matrix_a);

matrix_a.Cholesky(matrix_l);
Print("matrix_l\n", matrix_l);
Print("check\n", matrix_l.MatMul(matrix_l.Transpose()));

/*
matrix_a
[[5.7998084,-2.1825367]
 [-2.1825367,9.85910595]]
matrix_l
[[2.408279136645086,0]
 [-0.9062640068544704,3.006291985133859]]
check
[[5.7998084,-2.1825367]
 [-2.1825367,9.85910595]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Eig

Calcula los valores propios y los vectores propios derechos de una matriz cuadrada.

```
bool matrix::Eig(
  matrix&  eigen_vectors,         // matriz de vectores propios
  vector&  eigen_values           // vector de valores propios
   );
```

Es una solución compleja de valores propios y vectores propios

```
bool matrix::Eig(
  matrix<complex>& eigen_vectors, // matriz de vectores propios
  vector<complex>& eigen_values   // vector de valores propios
   );
```

Parámetros

eigen_vectors

[out] Matriz de vectores verticales propios.

eigen_values

[out] Vector de valores propios.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

Si se encuentra una solución compleja mientras se calculan los valores propios, el cálculo se detendrá, mostrando además el código de error 4019 (ERR_MATH_OVERFLOW). Para obtener la solución completa en unn espacio complejo, deberemos utilizar la sobrecarga compleja del método Eig.

Si un valor propio complejo tiene una parte imaginaria igual a cero, entonces ya será un valor propio real. Y esto se puede ver en el siguiente ejemplo.

Ejemplo:

```
void OnStart()
  {
   matrix matrix_a =
     {
        {-3.474589, 1.106384, -9.091977,-3.925227 },
        {-5.522139, 2.366887,-15.162351,-6.357512 },
        { 8.394926,-2.960067, 22.292115, 9.524129 },
        { 7.803242,-2.080287, 19.217706, 8.186645 }
     };

   matrix eigen_vectors;
   vector eigen_values;
```

© 2000-2025, MetaQuotes Ltd.

```
   bool res=matrix_a.Eig(eigen_vectors, eigen_values);
   Print("res=",res,"  error=",GetLastError());
   Print("Eigen vectors:\n",eigen_vectors, "\nEigen Values:\n",eigen_values);

//--- check correctness A * v = lambda * v
   int vectors=0;
   for(ulong n=0; n<eigen_values.Size(); n++)
     {
      vector eigen_vector=eigen_vectors.Col(n);
      vector vector_d1   =eigen_vector*eigen_values[n];
      vector vector_d2   =matrix_a.MatMul(eigen_vector);

      ulong errors=vector_d1.Compare(vector_d2,1e-13);
      if(errors==0)
         vectors++;
     }
   Print("vectors=",vectors);

//--- complex solution
   matrix<complex> eigen_vectors_c;
   vector<complex> eigen_values_c;
   ResetLastError();
   res=matrix_a.Eig(eigen_vectors_c,eigen_values_c);
   Print("res=",res,"  error=",GetLastError());
   Print("Eigen vectors:\n",eigen_vectors_c, "\nEigen Values:\n",eigen_values_c);

//--- check correctness A * v = lambda * v
   matrixc matrix_c;
   matrix_c.Assign(matrix_a);
   vectors=0;
   for(ulong n=0; n<eigen_values_c.Size(); n++)
     {
      vectorc eigen_vector_c=eigen_vectors_c.Col(n);
      vectorc vector_c1     =eigen_vector_c*eigen_values_c[n];
      vectorc vector_c2     =matrix_c.MatMul(eigen_vector_c);

      ulong errors=vector_c1.Compare(vector_c2,1e-13);
      if(errors==0)
         vectors++;
     }
   Print("vectors=",vectors);
  }
/* Resultado
   res=true  error=4019
   Eigen vectors:
   [[0.2649667608713664]
    [0.4488818803991876]
    [-0.6527335897527492]
```

© 2000-2025, MetaQuotes Ltd.

```
    [-0.5497604331807768]]
   Eigen Values:
   [28.94158645962942]
   vectors=1
   res=true  error=0
   Eigen vectors:
   [[(0.2649667608713664,0),(0.2227392219204762,0.3745470492013296),(0.403285439089771
    [(0.4488818803991876,0),(-0.2613452530342438,-0.3685707727819327),(-0.396850612637
    [(-0.6527335897527492,0),(-0.3418479634708521,-0.3299830378162041),(-0.355302103111
    [(-0.5497604331807768,0),(0.523227993452828,0.3262508584080381),(0.351283559614366
   Eigen Values:
   [(28.94158645962942,0),(0.01022501104810897,0.02954980822331488),(0.010225011048108
   vectors=3
*/
```

© 2000-2025, MetaQuotes Ltd.

EigVals

Calcula los valores propios de la matriz global.

```
bool matrix::EigVals(
  vector&  eigen_values      // vector de valores propios
   );
```

Parámetros

eigen_values

[out] Vector de valores propios derechos.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

La única diferencia entre EigVals y Eig es que EigVals no calcula los vectores propios, solo los valores propios.

© 2000-2025, MetaQuotes Ltd.

LU

Realiza una factorización LU de una matriz como producto de una matriz rectangular inferior y una matriz rectangular superior.

```
bool matrix::LU(
  matrix&  L,      // matriz triangular inferior
  matrix&  U       // matriz triangular superior
   );
```

Parámetros

L

[out] Matriz triangular inferior.

U

[out] Matriz singular superior.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Ejemplo:

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u;
//--- descomposición LU
   matrix_a.LU(matrix_l,matrix_u);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
//--- comprobamos la corrección de A = L * U
   Print("check\n",matrix_l.MatMul(matrix_u));


   /*
   matrix_l
   [[1,0,0,0]
    [5,1,0,0]
    [8,0.875,1,0]
    [11,1.25,0.5904761904761905,1]]
   matrix_u
   [[1,2,3,4]
    [0,-8,-9,-13]
    [0,0,-13.125,-10.625]
    [0,0,0,-17.47619047619047]]
   check
   [[1,2,3,4]
```

© 2000-2025, MetaQuotes Ltd.

```
   [5,2,6,7]
   [8,9,3,10]
   [11,12,14,4]]
  */
```

© 2000-2025, MetaQuotes Ltd.

LUP

Realiza una factorización LUP con rotación parcial, que es una factorización LU solo con permutaciones de líneas: PA=LU.

```
bool  LUP(
  matrix&  L,      // matriz triangular inferior
  matrix&  U,      // matriz triangular superior
  matrix&  P       // matriz de permutación
   );
```

Parámetros

L

[out] Matriz triangular inferior.

U

[out] Matriz singular superior.

P

[out] Matriz de permutación.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Ejemplo:

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u,matrix_p;
//--- descomposición LUP
   matrix_a.LUP(matrix_l,matrix_u,matrix_p);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
   Print("matrix_p\n",matrix_p);
//--- comprobamos la corrección de P * A = L * U
   Print("P * A\n",matrix_p.MatMul(matrix_a));
   Print("L * U\n",matrix_l.MatMul(matrix_u));

   /*
   matrix_l
   [[1,0,0,0]
    [0.4545454545454545,1,0,0]
    [0.7272727272727273,-0.07894736842105282,1,0]
    [0.09090909090909091,-0.2631578947368421,-0.2262773722627738,1]]
   matrix_u
   [[11,12,14,4]
```

© 2000-2025, MetaQuotes Ltd.

```
 [0,-3.454545454545454,-0.3636363636363633,5.181818181818182]
 [0,0,-7.210526315789473,7.500000000000001]
 [0,0,0,6.697080291970803]]
matrix_p
[[0,0,0,1]
 [0,1,0,0]
 [0,0,1,0]
 [1,0,0,0]]
P * A
[[11,12,14,4]
 [5,2,6,7]
 [8,9,3,10]
 [1,2,3,4]]
L * U
[[11,12,14,4]
 [5,2,6,7]
 [8,9,3.000000000000001,10]
 [1,2,3,4]]
*/
```

© 2000-2025, MetaQuotes Ltd.

QR

Calcula la factorización QR de una matriz.

```
bool  QR(
  matrix&  Q,      // matriz con columnas ortonormalizadas
  matrix&  R       // matriz triangular superior
   );
```

Parámetros

Q

[out] Matriz con columnas ortonormalizadas. Al activar el modo "complete", el resultado será una matriz ortogonal/unitaria, dependiendo de si A es real/compleja. En este caso, el determinante podría ser +/- 1. Si el número de dimensiones de la matriz de entrada es superior a 2, se retornará una pila de matrices con las propiedades anteriores.

R

[out] Matriz singular superior.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Ejemplo:

```
//---  A*x = b
  matrix A = {{0, 1}, {1, 1}, {1, 1}, {2, 1}};
  Print("A \n", A);
  vector b = {1, 2, 2, 3};
  Print("b \n", b);
//--- A = Q*R
  matrix q, r;
  A.QR(q, r);
  Print("q \n", q);
  Print("r \n", r);
  matrix qr=q.MatMul(r);
  Print("qr \n", qr);
  /*
  A
  [[0,1]
  [1,1]
  [1,1]
  [2,1]]
  b
  [1,2,2,3]
  q
  [[0.4082482904638631,-0.8164965809277259,-1.110223024625157e-16,-0.4082482904638631]
  [0.4625425214347352,-0.03745747856526496,0.7041241452319315,0.5374574785652647]
  [-0.5374574785652648,-0.03745747856526496,0.7041241452319316,-0.4625425214347352]
```

© 2000-2025, MetaQuotes Ltd.

```
[-0.5749149571305296,-0.5749149571305299,-0.09175170953613698,0.5749149571305296]]
r
[[-1.224744871391589,-0.2415816237971962]
[-1.22474487139159,-1.466326495188786]
[1.224744871391589,1.316496580927726]
[1.224744871391589,0.2415816237971961]]
qr
[[-1.110223024625157e-16,1]
[1,0.9999999999999999]
[1,1]
[2,1]]
*/
```

© 2000-2025, MetaQuotes Ltd.

SVD

Calcula la descomposición de valores singulares.

```
bool matrix::SVD(
  matrix&  U,                    // matriz unitaria
  matrix&  V,                    // matriz unitaria
  vector&  singular_values       // vector de valores singulares
   );
```

Parámetros

U

[out] Matriz unitaria de orden m, formada por vectores singulares izquierdos.

V

[out] Matriz unitaria de orden n, formada por vectores singulares derechos.

singular_values

[out] Valores singulares

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Ejemplo:

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
//--- realizamos una descomposición SVD
  matrix U, V;
  vector singular_values;
  b.SVD(U, V, singular_values);
  Print("U \n", U);
  Print("V \n", V);
  Print("singular_values = ", singular_values);

// bloque de verificación
//--- U * singular diagonal * V = A
  matrix matrix_s;
  matrix_s.Diag(singular_values);
  Print("matrix_s \n", matrix_s);
  matrix matrix_vt=V.Transpose();
  Print("matrix_vt \n", matrix_vt);
  matrix matrix_usvt=(U.MatMul(matrix_s)).MatMul(matrix_vt);
  Print("matrix_usvt \n", matrix_usvt);
```

© 2000-2025, MetaQuotes Ltd.

```
  ulong errors=(int)b.Compare(matrix_usvt, 1e-9);
  double res=(errors==0);
  Print("errors=", errors);

//---- otra verificación más
  matrix U_Ut=U.MatMul(U.Transpose());
  Print("U_Ut \n", U_Ut);
  Print("Ut_U \n", (U.Transpose()).MatMul(U));

  matrix vt_V=matrix_vt.MatMul(V);
  Print("vt_V \n", vt_V);
  Print("V_vt \n", V.MatMul(matrix_vt));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
  [[-4,-3,-2]
   [-1,0,1]
   [2,3,4]]
  U
  [[-0.7071067811865474,0.5773502691896254,0.408248290463863]
   [-6.827109697437648e-17,0.5773502691896253,-0.8164965809277256]
   [0.7071067811865472,0.5773502691896255,0.4082482904638627]]
  V
  [[0.5773502691896258,-0.7071067811865474,-0.408248290463863]
   [0.5773502691896258,1.779939029415334e-16,0.8164965809277258]
   [0.5773502691896256,0.7071067811865474,-0.408248290463863]]
  singular_values = [7.348469228349533,2.449489742783175,3.277709923350408e-17]

  matrix_s
  [[7.348469228349533,0,0]
   [0,2.449489742783175,0]
   [0,0,3.277709923350408e-17]]
  matrix_vt
  [[0.5773502691896258,0.5773502691896258,0.5773502691896256]
   [-0.7071067811865474,1.779939029415334e-16,0.7071067811865474]
   [-0.408248290463863,0.8164965809277258,-0.408248290463863]]
  matrix_usvt
  [[-3.999999999999997,-2.999999999999999,-2]
   [-0.9999999999999981,-5.977974170712231e-17,0.9999999999999974]
   [2,2.999999999999999,3.999999999999996]]
  errors=0

  U_Ut
  [[0.9999999999999993,-1.665334536937735e-16,-1.665334536937735e-16]
   [-1.665334536937735e-16,0.9999999999999987,-5.551115123125783e-17]
   [-1.665334536937735e-16,-5.551115123125783e-17,0.9999999999999999]]
```

© 2000-2025, MetaQuotes Ltd.

```
  Ut_U
  [[0.9999999999999993,-5.551115123125783e-17,-1.110223024625157e-16]
   [-5.551115123125783e-17,0.9999999999999987,2.498001805406602e-16]
   [-1.110223024625157e-16,2.498001805406602e-16,0.999999999999999]]
  vt_V
  [[1,-5.551115123125783e-17,0]
   [-5.551115123125783e-17,0.9999999999999996,1.110223024625157e-16]
   [0,1.110223024625157e-16,0.9999999999999996]]
  V_vt
  [[0.9999999999999999,1.110223024625157e-16,1.942890293094024e-16]
   [1.110223024625157e-16,0.9999999999999998,1.665334536937735e-16]
   [1.942890293094024e-16,1.665334536937735e-16,0.9999999999999996]
  */
 }
```

© 2000-2025, MetaQuotes Ltd.

Métodos estadísticos

Métodos para obtener estadísticas descriptivas de matrices y vectores.

Función	Acción
ArgMax	Retorna el índice del valor máximo
ArgMin	Retorna el índice del valor mínimo
Max	Retrona el valor máximo de la matriz/vector
Min	Retorna el valor mínimo de la matriz/vector
Ptp	Retorna el rango de valores de una matriz/vector o el eje de la matriz especificado
Sum	Suma los elementos de una matriz/vector, que también puede realizarse en un eje(s) especificado(s)
Prod	Calcula el producto de los elementos de la matriz/vector, que también podrá realizarse a lo largo de un eje especificado
CumSum	Devuelve la suma acumulativa (acumulada) de los elementos de la matriz/vector, incluidos los que se encuentran a lo largo de un eje determinado
CumProd	Devuelve el producto acumulativo de los elementos de la matriz/vector, incluidos los situados a lo largo de un eje determinado
Percentile	Calcula el percentil especificado de los valores de los elementos de la matriz/vector o los elementos a lo largo del eje especificado
Quantile	Calcula el cuantil especificado de los valores de los elementos de la matriz/vector o los elementos a lo largo del eje especificado
Median	Calcula la mediana de los elementos del array/vector
Mean	Calcula la media aritmética de los valores de los elementos
Average	Calcula la media ponderada de los valores de la matriz/vector
Std	Calcula la desviación media cuadrada (estándar) de los valores de los elementos de la matriz/vector o del eje especificado
Var	Calcula la varianza de los valores de los elementos de la matriz/vector
LinearRegression	Calcula un vector/matriz con los valores de regresión lineal calculados

© 2000-2025, MetaQuotes Ltd.

ArgMax

Retorna el índice del valor máximo.

```
ulong vector::ArgMax();

ulong matrix::ArgMax();

vector matrix::ArgMax(
  const int  axis      // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Índice del valor máximo.

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.ArgMax(0);
   vector rows_max=matrix_a.ArgMax(1);
   ulong  matrix_max=matrix_a.ArgMax();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max index ",matrix_max,"  max value ",matrix_a.Flat(matrix_max));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[0,3,1]
   rows_max=[0,2,0,1]
   max index 5  max value 12.0
   */
```

© 2000-2025, MetaQuotes Ltd.

ArgMin

Retorna el índice del valor mínimo.

```
ulong vector::ArgMin();

ulong matrix::ArgMin();

vector matrix::ArgMin(
  const int  axis      // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Índice del valor mínimo.

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.ArgMin(0);
   vector rows_min=matrix_a.ArgMin(1);
   ulong  matrix_min=matrix_a.ArgMin();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min index ",matrix_min,"  min value ",matrix_a.Flat(matrix_min));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,0,0]
   rows_min=[2,0,2,0]
   min index 3  min value 1.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Max

Retrona el valor máximo de la matriz/vector.

```
double vector::Max();

double matrix::Max();

vector matrix::Max(
  const int  axis      // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Retrona el valor máximo en la matriz/vector.

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.Max(0);
   vector rows_max=matrix_a.Max(1);
   double matrix_max=matrix_a.Max();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max value ",matrix_max);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[10,11,12]
   rows_max=[10,12,6,11]
   max value 12.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Min

Retorna el valor mínimo de la matriz/vector.

```
double vector::Min();

double matrix::Min();

vector matrix::Min(
  const int  axis      // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Valor mínimo en la matriz/vector.

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.Min(0);
   vector rows_min=matrix_a.Min(1);
   double matrix_min=matrix_a.Min();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min value ",matrix_min);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,3,2]
   rows_min=[2,1,4,7]
   min value 1.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Ptp

Retorna el rango de valores de la matriz/vector o el eje de la matriz especificado, y es equivalente al resultado Max() - Min(). Ptp - Peak to peak, de pico a pico.

```
double vector::Ptp();

double matrix::Ptp();

vector matrix::Ptp(
  const int  axis      // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Vector con rangos de valores (máximo - mínimo).

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_ptp=matrix_a.Ptp(0);
   vector rows_ptp=matrix_a.Ptp(1);
   double matrix_ptp=matrix_a.Ptp();

   Print("cols_ptp  ",cols_ptp);
   Print("rows_ptp  ",rows_ptp);
   Print("ptp value  ",matrix_ptp);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_ptp [9,8,10]
   rows_ptp [8,11,2,4]
   ptp value  11.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Sum

Suma los elementos de una matriz/vector, que también puede realizarse en un eje(s) especificado(s).

```
double vector::Sum();

double matrix::Sum();

vector matrix::Sum(
  const int  axis      // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Suma de los elementos de una matriz/vector, que también puede realizarse en un eje(s) especificado(s).

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_sum=matrix_a.Sum(0);
   vector rows_sum=matrix_a.Sum(1);
   double matrix_sum=matrix_a.Sum();

   Print("cols_sum=",cols_sum);
   Print("rows_sum=",rows_sum);
   Print("sum value ",matrix_sum);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_sum=[24,27,27]
   rows_sum=[15,21,15,27]
   sum value 78.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Prod

Calcula el producto de los elementos de la matriz/vector, que también podrá realizarse a lo largo de un eje especificado.

```
double vector::Prod(
  const double  initial=1      // multiplicador inicial
   );

double matrix::Prod(
  const double  initial=1      // multiplicador inicial
   );

vector matrix::Prod(
  const int     axis,          // eje
  const double  initial=1      // multiplicador inicial
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

initial=1

[in] Multiplicador inicial.

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_prod=matrix_a.Prod(0);
   vector rows_prod=matrix_a.Prod(1);
   double matrix_prod=matrix_a.Prod();

   Print("cols_prod=",cols_prod);
   cols_prod=matrix_a.Prod(0,0.1);
   Print("cols_prod=",cols_prod);
   Print("rows_prod=",rows_prod);
   Print("prod value ",matrix_prod);

  /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_prod=[420,1320,864]
   cols_prod=[42,132,86.400000000000001]
```

© 2000-2025, MetaQuotes Ltd.

```
   rows_prod=[60,96,120,693]
   prod value 479001600.0
  */
```

© 2000-2025, MetaQuotes Ltd.

CumSum

Devuelve la suma acumulativa (acumulada) de los elementos de la matriz/vector, incluidos los que se encuentran a lo largo de un eje determinado.

```
vector vector::CumSum();

vector matrix::CumSum();

matrix matrix::CumSum(
  const int  axis      // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Suma acumulada de los elementos a lo largo de un eje determinado.

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumsum=matrix_a.CumSum(0);
   matrix rows_cumsum=matrix_a.CumSum(1);
   vector cumsum_values=matrix_a.CumSum();

   Print("cols_cumsum\n",cols_cumsum);
   Print("rows_cumsum\n",rows_cumsum);
   Print("cumsum values  ",cumsum_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumsum
   [[10,3,2]
    [11,11,14]
    [17,16,18]
    [24,27,27]]
   rows_cumsum
   [[10,13,15]
    [1,9,21]
```

© 2000-2025, MetaQuotes Ltd.

```
   [6,11,15]
   [7,18,27]]
  cumsum values  [10,13,15,16,24,36,42,47,51,58,69,78]
  */
```

© 2000-2025, MetaQuotes Ltd.

CumProd

Devuelve el producto acumulativo de los elementos de la matriz/vector, incluidos los situados a lo largo de un eje determinado.

```
vector vector::CumProd();

vector matrix::CumProd();

matrix matrix::CumProd(
  const int  axis      // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal para cada columna (es decir, por filas), 1 — eje vertical para cada fila (es decir, por columnas).

Valor retornado

Producto acumulado de los elementos a lo largo de un eje determinado.

Ejemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumprod=matrix_a.CumProd(0);
   matrix rows_cumprod=matrix_a.CumProd(1);
   vector cumprod_values=matrix_a.CumProd();

   Print("cols_cumprod\n",cols_cumprod);
   Print("rows_cumprod\n",rows_cumprod);
   Print("cumprod values  ",cumprod_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumprod
   [[10,3,2]
    [10,24,24]
    [60,120,96]
    [420,1320,864]]
   rows_cumprod
   [[10,30,60]
```

© 2000-2025, MetaQuotes Ltd.

```
    [1,8,96]
    [6,30,120]
    [7,77,693]]
   cumprod values  [10,30,60,60,480,5760,34560,172800,691200,4838400,53222400,47900160
   */
```

© 2000-2025, MetaQuotes Ltd.

Percentile

Calcula el percentil especificado de los valores de los elementos de la matriz/vector o los elementos a lo largo del eje especificado.

```
double vector::Percentile(
  const int  percent      //
   );

double matrix::Percentile(
  const int  percent      //
   );

vector matrix::Percentile(
  const int  percent,     //
  const int  axis         // eje
   );
```

Parámetros

percent

[in] Percentil calculado, deberá estar comprendido entre 0 y 100 inclusive.

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Percentil: escalar o vectorial.

Observación

Los valores válidos para el porcentaje están en el intervalo [0, 100]. Para calcular los percentiles se usa un algoritmo lineal. La secuencia deberá estar clasificada para calcular correctamente los percentiles.

Ejemplo:

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_percentile=matrix_a.Percentile(50,0);
   vectorf rows_percentile=matrix_a.Percentile(50,1);
   float matrix_percentile=matrix_a.Percentile(50);

   Print("cols_percentile ",cols_percentile);
   Print("rows_percentile ",rows_percentile);
   Print("percentile value  ",matrix_percentile);
```

© 2000-2025, MetaQuotes Ltd.

```
/*
matrix_a
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
cols_percentile [5.5,6.5,7.5]
rows_percentile [2,5,8,11]
percentile value  6.5
*/
```

© 2000-2025, MetaQuotes Ltd.

Quantile

Calcula el cuantil especificado de los valores de los elementos de la matriz/vector o los elementos a lo largo del eje especificado.

```
double vector::Quantile(
  const double quantile      // cuantil
   );

double matrix::Quantile(
  const double quantile      // cuantil
   );

vector matrix::Quantile(
  const double quantile,     // cuantil
  const int    axis          // eje
   );
```

Parámetros

quantile

[in] Cuantil calculado, deberá estar comprendido entre 0 y 100 inclusive.

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical

Valor retornado

Cuantil: escalar o vector.

Observación

El rango de valores del parámetro "quantile" toma valores en el intervalo [0, 1] Para calcular el cuantil se utiliza un algoritmo lineal. La secuencia deberá estar clasificada para calcular correctamente los cuantiles.

Ejemplo:

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_quantile=matrix_a.Quantile(0.5,0);
   vectorf rows_quantile=matrix_a.Quantile(0.5,1);
   float matrix_quantile=matrix_a.Quantile(0.5);

   Print("cols_quantile ",cols_quantile);
   Print("rows_quantile ",rows_quantile);
   Print("quantile value  ",matrix_quantile);

   /*
```

© 2000-2025, MetaQuotes Ltd.

```
matrix_a
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
cols_quantile [5.5,6.5,7.5]
rows_quantile [2,5,8,11]
quantile value  6.5
*/
```

© 2000-2025, MetaQuotes Ltd.

Median

Calcula la mediana de los elementos del array/vector.

```
double vector::Median();

double matrix::Median();

vector matrix::Median(
  const int  axis      // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Mediana: escalar o vectorial.

Observación

La mediana es un valor tal que exactamente la mitad de los elementos de la matriz/vector serán menores que él, y la otra mitad serán mayores. Lo mismo que Quantile(0.5) y Percentile(50). La secuencia deberá estar clasificada para calcular correctamente los percentiles.

Ejemplo:

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_median=matrix_a.Median(0);
   vectorf rows_median=matrix_a.Median(1);
   float matrix_median=matrix_a.Median();

   Print("cols_median ",cols_median);
   Print("rows_median ",rows_median);
   Print("median value  ",matrix_median);


   /*
   matrix_a
   [[1,2,3]
    [4,5,6]
    [7,8,9]
    [10,11,12]]
   cols_median [5.5,6.5,7.5]
```

© 2000-2025, MetaQuotes Ltd.

```
   rows_median [2,5,8,11]
   median value  6.5
   */
```

© 2000-2025, MetaQuotes Ltd.

Mean

Calcula la media aritmética de los valores de los elementos.

```
double vector::Mean();

double matrix::Mean();

vector matrix::Mean(
  const int  axis      // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Media aritmética de los valores de los elementos.

Ejemplo:

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_mean=matrix_a.Mean(0);
   vectorf rows_mean=matrix_a.Mean(1);
   float matrix_mean=matrix_a.Mean();

   Print("cols_mean ",cols_mean);
   Print("rows_mean ",rows_mean);
   Print("mean value  ",matrix_mean);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_mean [6,6.75,6.75]
   rows_mean [5,7,5,9]
   mean value  6.5
   */
```

© 2000-2025, MetaQuotes Ltd.

Average

Calcula la media ponderada de los valores de la matriz/vector.

```
double vector::Average(
  const vector& weigts      // vector de pesos
   );

double matrix::Average(
  const matrix& weigts      // matriz de pesos
);

vector matrix::Average(
  const matrix& weigts,     // matriz de pesos
  const int     axis        // eje
   );
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

Media aritmética: escalar o vectorial.

Observación

La matriz/vector de ponderación se asocia a la matriz/vector principal.

Ejemplo:

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   matrixf matrix_w=matrixf::Ones(4,3);
   Print("matrix_a\n",matrix_a);

   vectorf cols_average=matrix_a.Average(matrix_w,0);
   vectorf rows_average=matrix_a.Average(matrix_w,1);
   float matrix_average=matrix_a.Average(matrix_w);

   Print("cols_average ",cols_average);
   Print("rows_average ",rows_average);
   Print("average value  ",matrix_average);


   /*
   matrix_a
   [[10,3,2]
```

© 2000-2025, MetaQuotes Ltd.

```
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_average [6,6.75,6.75]
   rows_average [5,7,5,9]
   average value  6.5
   */ value  6.5
```

© 2000-2025, MetaQuotes Ltd.

Std

Calcula la desviación media cuadrada (estándar) de los valores de los elementos de la matriz/vector o del eje especificado.

```
double vector::Std();

double vector::Std(
  const int  ddof      //  desplazamiento del grado de libertad
);

double matrix::Std();

vector matrix::Std(
  const int  axis      // eje
   );

vector matrix::Std(
  const int  axis,     // eje
  const int  ddof      //  desplazamiento del grado de libertad
);
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

ddof

[in] "Delta Degrees of Freedom": parámetro que determina cuánto disminuye el número de grados de libertad. El divisor en la fórmula será N - ddof, donde N será el número de observaciones. Por defecto ddof = 0.

Valor retornado

Desviación estándar: escalar o vectorial.

Observación

La desviación estándar es la raíz cuadrada de la media de los cuadrados de las desviaciones de la media, es decir, std = sqrt(mean(x)), donde x = abs(a - a.mean())**2.

La desviación estándar suele calcularse como x.sum() / N, donde N = len(x).

La expresión con *ddof*=0 a veces se llama "varianza del conjunto general". Si *ddof* > 0 (el valor se usará con mayor frecuencia 1), el valor resultante se llamará "varianza selectiva".

Ejemplo:

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);
```

© 2000-2025, MetaQuotes Ltd.

```
vectorf cols_std=matrix_a.Std(0);
vectorf rows_std=matrix_a.Std(1);
float matrix_std=matrix_a.Std();

Print("cols_std ",cols_std);
Print("rows_std ",rows_std);
Print("std value  ",matrix_std);


/*
matrix_a
[[10,3,2]
 [1,8,12]
 [6,5,4]
 [7,11,9]]
cols_std [3.2403703,3.0310888,3.9607449]
rows_std [3.5590262,4.5460606,0.81649661,1.6329932]
std value  3.452052593231201
*/
```

© 2000-2025, MetaQuotes Ltd.

Var

Calcula la varianza de los valores de los elementos de la matriz/vector.

```
double vector::Var();

double vector::Std(
  const int  ddof      //  desplazamiento del grado de libertad
);

double matrix::Var();

vector matrix::Var(
  const int  axis      // eje
   );

vector matrix::Std(
  const int  axis,     // eje
  const int  ddof      //  desplazamiento del grado de libertad
);
```

Parámetros

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

ddof

[in] "Delta Degrees of Freedom": parámetro que determina cuánto disminuye el número de grados de libertad. El divisor en la fórmula será N - ddof, donde N será el número de observaciones. Por defecto ddof = 0.

Valor retornado

Dispersión: escalar o vectorial.

Observación

La varianza es la desviación media cuadrada respecto a la media, es decir, var = mean(x), donde x = abs(a - a.mean())**2.

El valor medio suele calcularse como x.sum() / N, donde N = len(x).

La expresión con *ddof*=0 a veces se llama "varianza del conjunto general". Si *ddof* > 0 (el valor se usará con mayor frecuencia 1), el valor resultante se llamará "varianza selectiva".

Ejemplo:

```
matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
Print("matrix_a\n",matrix_a);
```

© 2000-2025, MetaQuotes Ltd.

```
vectorf cols_var=matrix_a.Var(0);
vectorf rows_var=matrix_a.Var(1);
float matrix_var=matrix_a.Var();

Print("cols_var ",cols_var);
Print("rows_var ",rows_var);
Print("var value  ",matrix_var);


/*
matrix_a
[[10,3,2]
 [1,8,12]
 [6,5,4]
 [7,11,9]]
cols_var [10.5,9.1875,15.6875]
rows_var [12.666667,20.666666,0.66666669,2.6666667]
var value  11.916666984558105
*/
```

© 2000-2025, MetaQuotes Ltd.

LinearRegression

Calcula un vector/matriz con los valores de regresión lineal calculados.

```
vector vector::LinearRegression();

matrix matrix::LinearRegression(
  ENUM_MATRIX_AXIS  axis=AXIS_NONE      // eje a lo largo del cual se calcula la regre
   );
```

Parámetros

axis

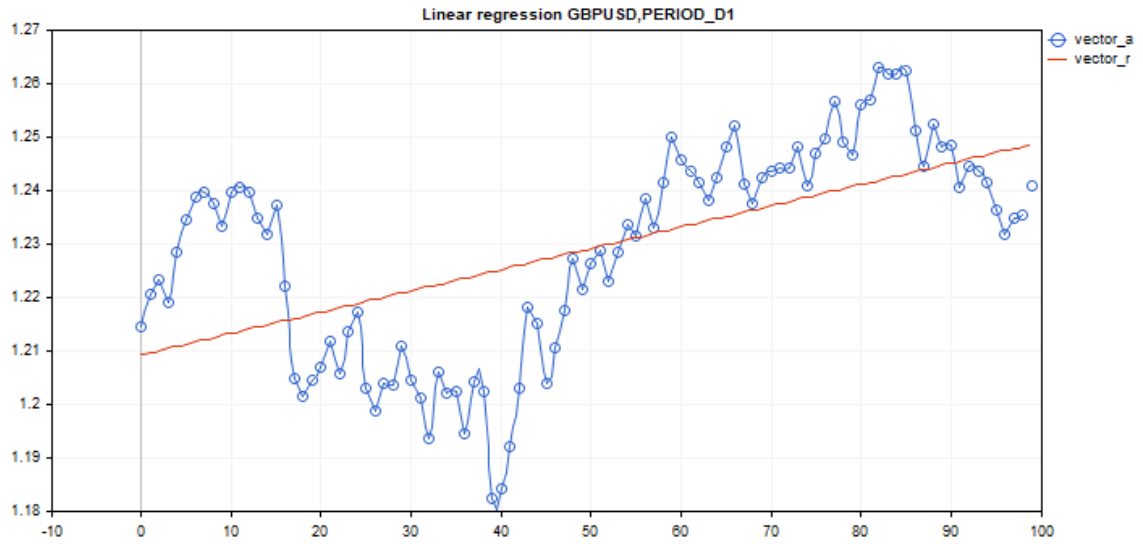
[in] Indicación del eje a lo largo del cual se calcula la regresión. Valor de la enumeración ENUM_MATRIX_AXIS (AXIS_HORZ — eje horizontal, AXIS_VERT — eje vertical).

Valor retornado

Vector o matriz con los valores de regresión lineal calculados.

Observación

Para calcular una regresión lineal, se utiliza una ecuación de regresión estándar: y (x) = a * x + b, donde a es la inclinación de la recta y b es su desplazamiento a lo largo del eje Y.



Ejemplo:

```
#include <Graphics\Graphic.mqh>

#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector vector_a;
   vector_a.CopyRates(_Symbol,_Period,COPY_RATES_CLOSE,1,100);
   vector vector_r=vector_a.LinearRegression();

//--- switch off chart show
   ChartSetInteger(0,CHART_SHOW,false);

//--- arrays for drawing a graph
   double x[];
   double y1[];
   double y2[];
   ArrayResize(x,uint(vector_a.Size()));
   ArrayResize(y1,uint(vector_a.Size()));
   ArrayResize(y2,uint(vector_a.Size()));
   for(ulong i=0; i<vector_a.Size(); i++)
     {
      x[i]=(double)i;
      y1[i]=vector_a[i];
      y2[i]=vector_r[i];
     }

//--- graph title
   string title="Linear regression "+_Symbol+","+EnumToString(_Period);

   long   chart=0;
   string name="LinearRegression";

//--- create graph
   CGraphic graphic;
   graphic.Create(chart,name,0,0,0,GRAPH_WIDTH,GRAPH_HEIGHT);
   graphic.BackgroundMain(title);
   graphic.BackgroundMainSize(12);

//--- activation function graph
   CCurve *curvef=graphic.CurveAdd(x,y1,CURVE_POINTS_AND_LINES);
   curvef.Name("vector_a");
   curvef.LinesWidth(2);
   curvef.LinesSmooth(true);
   curvef.LinesSmoothTension(1);
   curvef.LinesSmoothStep(10);

//--- derivatives of activation function
   CCurve *curved=graphic.CurveAdd(x,y2,CURVE_LINES);
   curved.Name("vector_r");
   curved.LinesWidth(2);
```

© 2000-2025, MetaQuotes Ltd.

```
   curved.LinesSmooth(true);
   curved.LinesSmoothTension(1);
   curved.LinesSmoothStep(10);
   graphic.CurvePlotAll();
   graphic.Update();

//--- endless loop to recognize pressed keyboard buttons
   while(!IsStopped())
     {
      //--- press escape button to quit program
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
         break;
      //--- press PdDn to save graph picture
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0)
        {
         string file_names[];
         if(FileSelectDialog("Save Picture",NULL,"All files (*.*)|*.*",FSD_WRITE_FILE,
            continue;
         ChartScreenShot(0,file_names[0],GRAPH_WIDTH,GRAPH_HEIGHT);
        }
      Sleep(10);
     }

//--- clean up
   graphic.Destroy();
   ObjectDelete(chart,name);
   ChartSetInteger(0,CHART_SHOW,true);
  }
```

ENUM_MATRIX_AXIS

Enumeración para indicar el eje en todas las funciones estadísticas para las matrices.

Identificador	Descripción
AXIS_NONE	El eje no está definido, el cálculo se realiza sobre todos los elementos de la matriz, como si se tratara de un vector (véase el método Flat).
AXIS_HORZ	Eje horizontal
AXIS_VERT	Eje vertical

© 2000-2025, MetaQuotes Ltd.

Métodos de características

Métodos para obtener las siguientes características de la matriz:

- número de filas
- número de columnas
- norma
- número de condiciones
- determinante
- rango de la matriz
- traza
- espectro

Función	Acción
Rows	Retorna el número de filas en la matriz
Cols	Retorna el número de columnas en la matriz
Size	Retorna el tamaño del vector
Norm	Retorna la norma de una matriz o vector
Cond	Calcula el número de condiciones de una matriz
Det	Calcula el determinante de una matriz cuadrada no degenerada
SLogDet	Calcula el signo y el logaritmo del determinante de la matriz
Rank	Retorna el rango de una matriz utilizando el método de Gauss
Trace	Retorna la suma de los elementos en las diagonales de la matriz
Spectrum	Retorna el espectro de una matriz como el conjunto de sus valores propios partiendo del producto AT*A

© 2000-2025, MetaQuotes Ltd.

Rows

Retorna el número de filas en la matriz.

```
ulong matrix::Rows()
```

Valor retornado

Valor del tipo integer.

Ejemplo:

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Rows()=", m.Rows());
Print("m.Cols()=", m.Cols());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Rows()=3
 m.Cols()=4
*/
```

© 2000-2025, MetaQuotes Ltd.

Cols

Retorna el número de columnas en la matriz.

```
ulong matrix::Cols()
```

Valor retornado

Valor del tipo integer.

Ejemplo:

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Cols()=", m.Cols());
Print("m.Rows()=", m.Rows());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Cols()=4
 m.Rows()=3
*/
```

© 2000-2025, MetaQuotes Ltd.

Size

Retorna el tamaño del vector.

```
ulong vector::Size()
```

Valor retornado

Valor del tipo integer.

Ejemplo:

```
matrix m={{1,2,3,4,5,6,7,8,9,10,11,12}};
m.Reshape(3,4);
Print("matrix m\n",m);
vector v=m.Row(1);
Print("v.Size()=",v.Size());
Print("v=",v);


/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 v.Size()=4
 v=[5,6,7,8]
*/
```

© 2000-2025, MetaQuotes Ltd.

Norm

Retorna la norma de una matriz o vector.

```
double vector::Norm(
  const ENUM_VECTOR_NORM  norm,     // norma del vector
  const int               norm_p=2  // número p-norm en el caso VECTOR_NORM_P
   );

double matrix::Norm(
  const ENUM_MATRIX_NORM  norm      // norma de la matriz
   );
```

Parámetros

norm

[in] Orden de la norma

Valor retornado

Norma de una matriz o vector.

Observación

- VECTOR_NORM_INF — valor absoluto máximo entre los elementos del vector.
- VECTOR_NORM_MINUS_INF — valor absoluto mínimo del vector.
- VECTOR_NORM_P — norma P del vector. Si norm_p=0, será el número de elementos del vector distintos de cero; norm_p=1 será la suma de los valores absolutos de los elementos del vector; norm_p=2 será la raíz cuadrada de la suma de los cuadrados de los valores de los elementos del vector. La norma P puede ser negativa.
- MATRIX_NORM_FROBENIUS — raíz cuadrada de la suma de los cuadrados de los valores de los elementos de la matriz. La norma de Frobenius y la norma P2 del vector son coherentes.
- MATRIX_NORM_SPECTRAL — valor máximo del espectro de la matriz.
- MATRIX_NORM_NUCLEAR — suma de los valores singulares de la matriz.
- MATRIX_NORM_INF — norma P1 vectorial máxima entre los vectores verticales de la matriz. La norma inf de la matriz y la norma del vector son consistentes.
- MATRIX_NORM_MINUS_INF — norma P1 vectorial mínima entre los vectores verticales de la matriz.
- MATRIX_NORM_P1 — norma P1 vectorial máxima entre los vectores horizontales de la matriz.
- MATRIX_NORM_MINUS_P1 — norma P1 vectorial mínima entre los vectores horizontales de la matriz.
- MATRIX_NORM_P2 — valor singular mayor de la matriz.
- MATRIX_NORM_MINUS_P2 — valor singular menor de la matriz.

Aquí vemos un algoritmo simple para calcular el vector de norma P en MQL5:

```
double VectorNormP(const vector& v,int norm_value)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   ulong  i;
   double norm=0.0;
//---
   switch(norm_value)
     {
      case 0 :
         for(i=0; i<v.Size(); i++)
            if(v[i]!=0)
               norm+=1.0;
         break;
      case 1 :
         for(i=0; i<v.Size(); i++)
            norm+=MathAbs(v[i]);
         break;
      case 2 :
         for(i=0; i<v.Size(); i++)
            norm+=v[i]*v[i];
         norm=MathSqrt(norm);
         break;
      default :
         for(i=0; i<v.Size(); i++)
            norm+=MathPow(MathAbs(v[i]),norm_value);
         norm=MathPow(norm,1.0/norm_value);
     }
//---
   return(norm);
  }
```

Ejemplo en MQL5:

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_FROBENIUS)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_INF)", b.Norm(MATRIX_NORM_INF));
  Print("b.Norm(MATRIX_NORM_MINUS_INF)", b.Norm(MATRIX_NORM_MINUS_INF));
  Print("b.Norm(MATRIX_NORM_P1)=)", b.Norm(MATRIX_NORM_P1));
  Print("b.Norm(MATRIX_NORM_MINUS_P1)=", b.Norm(MATRIX_NORM_MINUS_P1));
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_P2));
  Print("b.Norm(MATRIX_NORM_MINUS_P2)=", b.Norm(MATRIX_NORM_MINUS_P2));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
```

© 2000-2025, MetaQuotes Ltd.

```
  [[-4,-3,-2]
  [-1,0,1]
  [2,3,4]]
  b.Norm(MATRIX_NORM_P2)=7.745966692414834
  b.Norm(MATRIX_NORM_FROBENIUS)=7.745966692414834
  b.Norm(MATRIX_NORM_INF)9.0
  b.Norm(MATRIX_NORM_MINUS_INF)2.0
  b.Norm(MATRIX_NORM_P1)=)7.0
  b.Norm(MATRIX_NORM_MINUS_P1)=6.0
  b.Norm(MATRIX_NORM_P2)=7.348469228349533
  b.Norm(MATRIX_NORM_MINUS_P2)=1.857033188519056e-16
  */
```

Ejemplo en Python:

```
import numpy as np
from numpy import linalg as LA
a = np.arange(9) - 4
print("a \n",a)
b = a.reshape((3, 3))
print("b \n",b)
print("LA.norm(b)=",LA.norm(b))
print("LA.norm(b, 'fro')=",LA.norm(b, 'fro'))
print("LA.norm(b, np.inf)=",LA.norm(b, np.inf))
print("LA.norm(b, -np.inf)=",LA.norm(b, -np.inf))
print("LA.norm(b, 1)=",LA.norm(b, 1))
print("LA.norm(b, -1)=",LA.norm(b, -1))
print("LA.norm(b, 2)=",LA.norm(b, 2))
print("LA.norm(b, -2)=",LA.norm(b, -2))

a
 [-4 -3 -2 -1  0  1  2  3  4]
b
 [[-4 -3 -2]
 [-1  0  1]
 [ 2  3  4]]
LA.norm(b)= 7.745966692414834
LA.norm(b, 'fro')= 7.745966692414834
LA.norm(b, np.inf)= 9.0
LA.norm(b, -np.inf)= 2.0
LA.norm(b, 1)= 7.0
LA.norm(b, -1)= 6.0
LA.norm(b, 2)= 7.3484692283495345
LA.norm(b, -2)= 1.857033188519056e-16
```

© 2000-2025, MetaQuotes Ltd.

Cond

Calcula el número de condiciones de una matriz.

```
double matrix::Cond(
  const ENUM_MATRIX_NORM  norm      // norma de la matriz
   );
```

Parámetros

norm

[in] Orden de la norma de la enumeración ENUM_MATRIX_NORM.

Valor retornado

Número de condiciones de una matriz. Puede ser infinito.

Observación

El número de condición x se define como el producto de la norma x y su inversa x [1]. La norma puede ser la norma L2 habitual (la raíz de la suma de los cuadrados) o una de las otras normas matriciales.

El número (o medida) de la condición de una matriz se denomina K, y es igual al producto de la norma de la matriz A y su inversa. Las matrices con un gran número de condiciones se denominan mal condicionadas, y viceversa, las matrices con un número pequeño se denominan bien condicionadas. La matriz inversa se obtiene por pseudoinversión para no verse limitado por la condición de cuadratura y no degeneración de la matriz.

La excepción es el número de condición espectral.

Aquí vemos un algoritmo sencillo para calcular el número de condición espectral en MQL5:

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

Ejemplo en MQL5:

```
matrix a= {{1, 0, -1}, {0, 1, 0}, { 1, 0, 1}};
Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
Print("a.Cond(MATRIX_NORM_FROBENIUS)=", a.Cond(MATRIX_NORM_FROBENIUS));
Print("a.Cond(MATRIX_NORM_INF)=", a.Cond(MATRIX_NORM_INF));
Print("a.Cond(MATRIX_NORM_MINUS_INF)=", a.Cond(MATRIX_NORM_MINUS_INF));
Print("a.Cond(MATRIX_NORM_P1)=)", a.Cond(MATRIX_NORM_P1));
Print("a.Cond(MATRIX_NORM_MINUS_P1)=", a.Cond(MATRIX_NORM_MINUS_P1));
Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
Print("a.Cond(MATRIX_NORM_MINUS_P2)=", a.Cond(MATRIX_NORM_MINUS_P2));

/*
matrix a
[[1,0,-1]
[0,1,0]
[1,0,1]]
a.Cond(MATRIX_NORM_P2)=1.414213562373095
a.Cond(MATRIX_NORM_FROBENIUS)=3.162277660168379
a.Cond(MATRIX_NORM_INF)=2.0
a.Cond(MATRIX_NORM_MINUS_INF)=0.9999999999999997
a.Cond(MATRIX_NORM_P1)=)2.0
a.Cond(MATRIX_NORM_MINUS_P1)=0.9999999999999998
a.Cond(MATRIX_NORM_P2)=1.414213562373095
a.Cond(MATRIX_NORM_MINUS_P2)=0.7071067811865472
*/
```

Ejemplo en Python:

```
import numpy as np
from numpy import linalg as LA
a = np.array([[1, 0, -1], [0, 1, 0], [1, 0, 1]])
print("a \n",a)
print("LA.cond(a)=",LA.cond(a))
print("LA.cond(a, 'fro')=",LA.cond(a, 'fro'))
print("LA.cond(a, np.inf)=",LA.cond(a, np.inf))
print("LA.cond(a, -np.inf)=",LA.cond(a, -np.inf))
```

© 2000-2025, MetaQuotes Ltd.

```
print("LA.cond(a, 1)=",LA.cond(a, 1))
print("LA.cond(a, -1)=",LA.cond(a, -1))
print("LA.cond(a, 2)=",LA.cond(a, 2))
print("LA.cond(a, -2)=",LA.cond(a, -2))

a
 [[ 1  0 -1]
 [ 0  1  0]
 [ 1  0  1]]
LA.cond(a)= 1.4142135623730951
LA.cond(a, 'fro')= 3.1622776601683795
LA.cond(a, np.inf)= 2.0
LA.cond(a, -np.inf)= 1.0
LA.cond(a, 1)= 2.0
LA.cond(a, -1)= 1.0
LA.cond(a, 2)= 1.4142135623730951
LA.cond(a, -2)= 0.7071067811865475
```

© 2000-2025, MetaQuotes Ltd.

Det

Calcula el determinante de una matriz cuadrada no degenerada.

```
double matrix::Det()
```

Valor retornado

Determinante de la matriz.

Observación

Los determinantes de las matrices de órdenes 2 y 3 se calculan usando la regla de Sarrus. d2=a11*a22-a12*a21; d3=a11*a22*a33+a12*a23*a31+a13*a21*a32-a13*a22*a31-a11*a23*a32-a12*a21*a33

El determinante se calcula usando el método de Gauss, convirtiendo la matriz a una forma no triangular superior. El determinante de una matriz no triangular superior es igual al producto de los miembros de la diagonal principal.

Si al menos una fila (columna) de una matriz es igual a cero, el determinante será cero.

Si dos (o más) filas (o columnas) de una matriz son linealmente dependientes, entonces su determinante será cero.

El determinante de una matriz es igual al producto de sus valores propios.

Ejemplo en MQL5:

```
matrix m={{1,2},{3,4}};
double det=m.Det();
Print("matrix m\n",m);
Print("det(m)=",det);
/*
matrix m
[[1,2]
 [3,4]]
det(m)=-2.0
*/
```

Ejemplo en Python:

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
print('a \n',a)
print('nnp.linalg.det(a) \n',np.linalg.det(a))

a
```

© 2000-2025, MetaQuotes Ltd.

```
 [[1 2]
 [3 4]]

np.linalg.det(a)
 -2.0000000000000004
```

© 2000-2025, MetaQuotes Ltd.

SLogDet

Calcula el signo y el logaritmo del determinante de la matriz.

```
double matrix::SLogDet(
  int&  sign      // signo
   );
```

Parámetros

sign

[out] Signo del determinante. Si el valor sign es par, el determinante será positivo.

Valor retornado

Número que representa el signo del determinante.

Observación

El determinante se calcula usando el método de Gauss, convirtiendo la matriz a una forma no triangular superior. El determinante de una matriz no triangular superior es igual al producto de los miembros de la diagonal principal. El logaritmo del producto será igual a la suma de los logaritmos. Por lo tanto, en caso de desbordamiento al calcular el determinante, se podrá utilizar el método SLogDet.

Si el valor sign es par, el determinante será positivo.

Ejemplo:

```
a = np.array([[1, 2], [3, 4]])  (sign, logdet) = np.linalg.slogdet(a)  (sign, logdet)
```

© 2000-2025, MetaQuotes Ltd.

Rank

Retorna el rango de una matriz utilizando el método de Gauss.

```
int  Rank()
```

Valor retornado

Rango de la matriz.

Observación

El rango de un sistema de filas (columnas) de una matriz A con m filas y n columnas es el número máximo de filas (columnas) linealmente independientes. Varias filas (columnas) se llaman linealmente independientes si ninguna de ellas se expresa linealmente a través de otras. El rango de un sistema de filas es siempre igual al rango del sistema de columnas, y este número se denomina rango de la matriz.

Ejemplo en MQL5:

```
matrix a=matrix::Eye(4, 4);;
Print("matrix a \n", a);
Print("a.Rank()=", a.Rank());

matrix I=matrix::Eye(4, 4);
I[3, 3] = 0.;    // déficit de la matriz
Print("I \n", I);
Print("I.Rank()=", I.Rank());

matrix b=matrix::Ones(1, 4);
Print("b \n", b);
Print("b.Rank()=", b.Rank());;// 1 dimensionalidad - rango 1, solo si no todos son '

matrix  zeros=matrix::Zeros(4, 1);
Print("zeros \n", zeros);
Print("zeros.Rank()=", zeros.Rank());

/*
matrix a
[[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[0,0,0,1]]
a.Rank()=4

I
[[1,0,0,0]
[0,1,0,0]
```

© 2000-2025, MetaQuotes Ltd.

```
  [0,0,1,0]
  [0,0,0,0]]
  I.Rank()=3

  b
  [[1,1,1,1]]
  b.Rank()=1

  zeros
  [[0]
  [0]
  [0]
  [0]]
  zeros.Rank()=0
  */
```

Ejemplo en Python:

```
import numpy as np
from numpy.linalg import matrix_rank
a=(np.eye(4)) # Full rank matrix
print("a \n", a)
print("matrix_rank(a)=",matrix_rank(a))
I=np.eye(4)
I[-1,-1] = 0. # rank deficient matrix
print("I \n",I)
print("matrix_rank(I)=",matrix_rank(I))

b=np.ones((4,))
print("b \n",b)
print("matrix_rank(b)=",matrix_rank(b)) # 1 dimension - rank 1 unless all 0

zeros=np.zeros((4,))
print("zeroes \n",zeros)
print("matrix_rank(zeros)=",matrix_rank(zeros))

a
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
matrix_rank(a)= 4

I
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]]
```

© 2000-2025, MetaQuotes Ltd.

```
matrix_rank(I)= 3

b
 [1. 1. 1. 1.]
matrix_rank(b)= 1

zeroes
 [0. 0. 0. 0.]
matrix_rank(zeros)= 0
```

© 2000-2025, MetaQuotes Ltd.

Trace

Retorna la suma de los elementos en las diagonales de la matriz.

```
double matrix::Trace()
```

Valor retornado

Suma de los elementos de la diagonal.

Observación

La traza de la matriz será igual a la suma de sus valores propios.

Ejemplo en MQL5:

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a.Reshape(3, 3);
  Print("matrix a \n", a);
  Print("a.Trace() \n", a.Trace());

  /*
  matrix a
  [[0,1,2]
  [3,4,5]
  [6,7,8]]
  a.Trace()
  12.0
  */
```

Ejemplo en Python:

```
a = np.arange(9).reshape((3,3))
print('a \n',a)
print('np.trace(a) \n',np.trace(a))

a
 [[0 1 2]
 [3 4 5]
 [6 7 8]]

np.trace(a)
 12
```

© 2000-2025, MetaQuotes Ltd.

Spectrum

Retorna el espectro de una matriz como el conjunto de sus valores propios partiendo del producto AT*A.

```
vector matrix::Spectrum()
```

Valor retornado

Espectro de una matriz como vector de valores propios de la matriz.

Ejemplo:

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
        }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

© 2000-2025, MetaQuotes Ltd.

Métodos matriciales para resolver sistemas de ecuaciones lineales

Métodos para resolver sistemas de ecuaciones lineales y calcular la matriz inversa.

Función	Acción
Solve	Resuelve una ecuación matricial lineal o un sistema de ecuaciones algebraicas lineales
LstSq	Resuelve aproximadamente un sistema de ecuaciones algebraicas lineales (para matrices no cuadradas o degeneradas)
Inv	Calcula una matriz inversa multiplicativa con respecto a una matriz cuadrada no degenerada utilizando el método de Jordaan-Gauss.
PInv	Calcula una matriz pseudoinversa según el método de Moore-Penrose

© 2000-2025, MetaQuotes Ltd.

Solve

Resuelve una ecuación matricial lineal o un sistema de ecuaciones algebraicas lineales.

```
vector matrix::Solve(
  const vector  b      // valores de la ordenada o "variable dependiente"
   );
```

Parámetros

b

[in] Valores de la ordenada o "variable dependiente". (Vector de miembros libres).

Valor retornado

Un vector con la solución del sistema a * x = b. * x = b.

Observación

Si al menos una fila (columna) de una matriz es cero, el sistema no tendrá solución.

Si dos (o más) filas (columnas) de una matriz son linealmente dependientes, el sistema no tendrá solución.

Ejemplo:

```
//--- solución de SEL
   vector_x=matrix_a.Solve(vector_b);
//--- comprobamos si a * x = b es correcto
   result_vector=matrix_a.MatMul(vector_x);
   errors=vector_b.Compare(result_vector,1e-12);
```

© 2000-2025, MetaQuotes Ltd.

LstSq

Resuelve aproximadamente un sistema de ecuaciones algebraicas lineales (para matrices no cuadradas o degeneradas).

```
vector matrix::LstSq(
  const vector  b      // valores de la ordenada o "variable dependiente"
   );
```

Parámetros

b

[in] Valores de la ordenada o "variable dependiente". (Vector de miembros libres)

Valor retornado

Un vector con la solución del sistema a * x = b. * x = b. Esto solo es cierto para los sistemas que tienen una solución exacta.

Ejemplo:

```
   matrix a={{3, 2},
             {4,-5},
             {3, 3}};
   vector b={7,40,3};
//---
   vector x=a.LstSq(b);
//--- comprobación, debe ser [5, -4]
   Print("x=", x);
//--- comprobación, debe ser [7, 40, 3]
   vector b1=a.MatMul(x);
   Print("b1=",b1);

/*
  x=[5.000000000000002,-4]
  b1=[7.000000000000005,40.00000000000001,3.000000000000005]
*/
```

© 2000-2025, MetaQuotes Ltd.

Inv

Calcula una matriz inversa multiplicativa con respecto a una matriz cuadrada no degenerada utilizando el método de Jordaan-Gauss.

```
matrix matrix::Inv()
```

Valor retornado

Matriz inversa multiplicativa.

Observación

El producto de la matriz origen y una matriz inversa será una matriz unitaria.

Si al menos una fila (columna) de una matriz es cero, no se podrá obtener la matriz inversa.

Si dos (o más) filas (columnas) de una matriz son linealmente dependientes, no se podrá obtener la matriz inversa.

Ejemplo:

```
int TestInverse(const int size_m)
  {
   int    i,j,errors=0;
   matrix matrix_a(size_m,size_m);
//--- rellenamos la matriz cuadrada
   MatrixTestFirst(matrix_a);
//--- mediremos los milisegundos
   ulong t1=GetMicrosecondCount();
//--- obtenemos la matriz inversa
   matrix inverse=matrix_a.Inv();
//--- medimos
   ulong t2=GetMicrosecondCount();
//--- comprobamos que sea correcto
   matrix identity=matrix_a.MatMul(inverse);
//---
   for(i=0; i<size_m; i++)
     {
      for(j=0; j<size_m; j++)
        {
         double value;
         //--- en la diagonal deberá haber unidades
         if(i==j)
            value=1.0;
         else
            value=0.0;
         if(MathClassify(identity[i][j])>FP_ZERO)
            errors++;
```

© 2000-2025, MetaQuotes Ltd.

```
         else
           {
            if(identity[i][j]!=value)
              {
               double diff=MathAbs(identity[i][j]-value);
               //--- muchas multiplicaciones y divisiones, por eso reducimos la precis
               if(diff>1e-9)
                  errors++;
              }
           }
        }
     }
//---
   double elapsed_time=double(t2-t1)/1000.0;
   printf("Inversion of matrix %d x %d  %s  errors=%d  time=%.3f ms",size_m,size_m,er
   return(errors);
  }
```

© 2000-2025, MetaQuotes Ltd.

PInv

Calcula una matriz pseudoinversa según el método de Moore-Penrose.

```
matrix matrix::PInv()
```

Valor retornado

Matriz pseudoinversa.

Ejemplo:

```
int TestPseudoInverse(const int size_m, const int size_k)
  {
   matrix matrix_a(size_m,size_k);
   matrix matrix_inverted(size_k,size_m);
   matrix matrix_temp;
   matrix matrix_a2;
//--- rellenamos la matriz
   MatrixTestFirst(matrix_a);
//--- invertimos
   matrix_inverted=matrix_a.PInv();
//--- comprobamos que sea correcto
   int errors=0;
//--- A * A+ * A = A   (A+ - matriz pseudoinversa de A)
   matrix_temp=matrix_a.MatMul(matrix_inverted);
   matrix_a2=matrix_temp.MatMul(matrix_a);
   errors=(int)matrix_a.CompareByDigits(matrix_a2,10);

   printf("PseudoInversion %s matrix_size  %d x %d  errors=%d",errors==0?"passed":"fai
//---
   return(errors);
  }
```

© 2000-2025, MetaQuotes Ltd.

Aprendizaje automático

Métodos para el aprendizaje automático.

La función de activación en una red neuronal determina el valor de salida de las neuronas según el resultado de la suma ponderada de las entradas. La elección de la función de activación influye sustancialmente en las capacidades y el rendimiento de la red neuronal. Las diferentes partes del modelo (capas) pueden utilizar funciones de activación distintas.

MQL5 implementa no solo todas las funciones de activación conocidas, sino también sus derivadas. Las derivadas de las funciones permiten actualizar eficazmente los parámetros del modelo según el error de aprendizaje.

La tarea de entrenar una red neuronal consiste en encontrar un algoritmo que minimice el error en la muestra de entrenamiento, usando para ello la función de pérdida. El valor de la función de pérdida describe la magnitud de desviación del valor predicho por el modelo respecto al valor real. Dependiendo del tipo de tarea a resolver se usarán diferentes funciones de pérdida. Por ejemplo: para un problema de regresión, será mean squared error (MSE), para la clasificación binaria, será binary cross-entropy (BCE)..

Función	Acción
Activation	Calcula los valores de la función de activación y los escribe en el vector/matriz trasmitido
Derivative	Calcula los valores de la derivada de la función de activación y los escribe en el vector/matriz transmitido
Loss	Calcula los valores de la función de pérdida y los escribe en el vector/matriz trasmitido
LossGradient	Calcula un vector o matriz de gradientes de la función de pérdida
RegressionMetric	Calcula la métrica de regresión como el error de desviación respecto a la línea de regresión trazada en el conjunto de datos especificado.
ConfusionMatrix	Calcula la matriz de error. El método se aplica a un vector de valores predichos
ConfusionMatrixMultilabel	Calcula la matriz de error para cada etiqueta. El método se aplica a un vector de valores predichos
ClassificationMetric	Calcula la métrica clasificatoria para evaluar la calidad de los datos previstos respecto a los datos reales. El método se aplica a un vector de valores predichos
ClassificationScore	Calcula la métrica clasificatoria para evaluar la calidad de los datos previstos respecto a los datos reales
PrecisionRecall	Calcula los valores para construir la curva precision-recall. Este método, al igual que el método ClassificationScore, se aplica al vector de valores verdaderos.

© 2000-2025, MetaQuotes Ltd.

Función	Acción
ReceiverOperatingCharacteristic	Calcula los valores para construir la curva Receiver Operating Characteristic (ROC). Este método, al igual que el método ClassificationScore, se aplica al vector de valores verdaderos.

Ejemplo:

Este ejemplo muestra el entrenamiento de un modelo usando operaciones matriciales. El modelo se entrena con la función (a + b + c)^2 / (a^2 + b^2 + c^2). La entrada es una matriz de datos de origen en las que a, b y c se encuentran en columnas aparte. La salida del modelo retorna el resultado de la función.

```
matrix weights1, weights2, weights3;                // matrices de coeficientes de pes
matrix output1, output2, result;                    // matrices de salidas de redes ne
input int layer1 = 200;                             // tamaño de la 1-era capa oculta
input int layer2 = 200;                             // tamaño de la 2-nda capa oculta
input int Epochs = 20000;                           // número de epocas de entrenamient
input double lr = 3e-6;                             // tasa de aprendizaje
input ENUM_ACTIVATION_FUNCTION ac_func = AF_SWISH; // función de activación
//+------------------------------------------------------------------+
//| Función de inicio del script                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int train = 1000;    // tamaño de la muestra de entrenamiento
   int test = 10;       // tamaño de la muestra de prueba
   matrix m_data, m_target;
//--- generamos la muestra de entrenamiento
   if(!CreateData(m_data, m_target, train))
      return;
//--- entrenamos el modelo
   if(!Train(m_data, m_target, Epochs))
      return;
//--- generamos una muestra de prueba
   if(!CreateData(m_data, m_target, test))
      return;
//--- probamos el modelo
   Test(m_data, m_target);
  }
//+------------------------------------------------------------------+
//| Método para generar la muestra                                   |
//+------------------------------------------------------------------+
bool CreateData(matrix &data, matrix &target, const int count)
  {
//--- inicializar las matrices de entrada y salida
   if(!data.Init(count, 3) || !target.Init(count, 1))
```

© 2000-2025, MetaQuotes Ltd.

```
      return false;
//--- rellenar la matriz de datos de entrada con valores aleatorios
   data.Random(-10, 10);
//--- calculamos los valores objetivo de la muestra de entrenamiento
   vector X1 = MathPow(data.Col(0) + data.Col(1) + data.Col(1), 2);
   vector X2 = MathPow(data.Col(0), 2) + MathPow(data.Col(1), 2) + MathPow(data.Col(2)
   if(!target.Col(X1 / X2, 0))
      return false;
//--- retornamos el resultado
   return true;
  }
//+------------------------------------------------------------------+
//| Método para entrenar el modelo                                   |
//+------------------------------------------------------------------+
bool Train(matrix &data, matrix &target, const int epochs = 10000)
  {
//--- creamos el modelo
   if(!CreateNet())
      return false;
//--- entrenamos el modelo
   for(int ep = 0; ep < epochs; ep++)
     {
      //--- pasada directa
      if(!FeedForward(data))
         return false;
      PrintFormat("Epoch %d, loss %.5f", ep, result.Loss(target, LOSS_MSE));
      //--- pasada inversa y actualización de las matrices de pesos
      if(!Backprop(data, target))
         return false;
     }
//--- retornamos el resultado
   return true;
  }
//+------------------------------------------------------------------+
//| Método para crear el modelo                                      |
//+------------------------------------------------------------------+
bool CreateNet()
  {
//--- inicializamos las matrices de pesos
   if(!weights1.Init(4, layer1) || !weights2.Init(layer1 + 1, layer2) || !weights3.Ini
      return false;
//--- rellenamos las matrices de pesos con valores aleatorios
   weights1.Random(-0.1, 0.1);
   weights2.Random(-0.1, 0.1);
   weights3.Random(-0.1, 0.1);
//--- retornamos el resultado
   return true;
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Método de pasada directa                                         |
//+------------------------------------------------------------------+
bool FeedForward(matrix &data)
  {
//--- comprobamos el tamaño de los datos de origen
   if(data.Cols() != weights1.Rows() - 1)
      return false;
//--- calculamos la primera capa neuronal
   matrix temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
   output1 = temp.MatMul(weights1);
//--- calculamos la función de activación
   if(!output1.Activation(temp, ac_func))
      return false;
//--- calculamos la segunda red neuronal
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   output2 = temp.MatMul(weights2);
//--- calculamos la función de activación
   if(!output2.Activation(temp, ac_func))
      return false;
//--- calculamos la tercera capa neuronal
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   result = temp.MatMul(weights3);
//--- retornamos el resultado
   return true;
  }
//+------------------------------------------------------------------+
//| Método de pasada inversa                                         |
//+------------------------------------------------------------------+
bool Backprop(matrix &data, matrix &target)
  {
//--- comprobamos la dimensionalidad de la matriz de valores objetivo
   if(target.Rows() != result.Rows() ||
      target.Cols() != result.Cols())
      return false;
//--- determinamos la desviación de los valores calculados respecto a los valores obje
   matrix loss = (target - result) * 2;
//--- gradiente hasta la capa anterior
   matrix gradient = loss.MatMul(weights3.Transpose());
//--- actualizamos la matriz de pesos de la última capa
   matrix temp;
   if(!output2.Activation(temp, ac_func))
      return false;
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   weights3 = weights3 + temp.Transpose().MatMul(loss) * lr;
//--- corregimos el gradiente de error usando la derivada de la función de activación
   if(!output2.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- bajamos el gradiente a una capa inferior
   gradient = loss.MatMul(weights2.Transpose());
//--- actualizamos la matriz de pesos de la 2-nda capa oculta
   if(!output1.Activation(temp, ac_func))
      return false;
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   weights2 = weights2 + temp.Transpose().MatMul(loss) * lr;
//--- corregimos el gradiente de error usando la derivada de la función de activación
   if(!output1.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- actualizamos la matriz de pesos de la 1-era capa oculta
   temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
   weights1 = weights1 + temp.Transpose().MatMul(loss) * lr;
//--- retornamos el resultado
   return true;
  }
//+------------------------------------------------------------------+
//| Método de prueba del modelo                                      |
//+------------------------------------------------------------------+
bool Test(matrix &data, matrix &target)
  {
//--- pasada directa con los datos de prueba
   if(!FeedForward(data))
      return false;
//--- mostramos en el log los resultados del cálculo del modelo y los valores reales
   PrintFormat("Test loss %.5f", result.Loss(target, LOSS_MSE));
   ulong total = data.Rows();
   for(ulong i = 0; i < total; i++)
      PrintFormat("(%.2f + %.2f + %.2f)^2 / (%.2f^2 + %.2f^2 + %.2f^2) =  Net %.2f, Ta
                  data[i, 0], data[i, 1], data[i, 2], result[i, 0], target[i, 0]);
//--- retornamos el resultado
```

© 2000-2025, MetaQuotes Ltd.

```
   return true;
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

Activation

Calcula los valores de la función de activación y los escribe en el vector/matriz trasmitido.

```
bool vector::Activation(
  vector&                   vect_out,      // vector para obtener los valores
  ENUM_ACTIVATION_FUNCTION  activation,    // función de activación
   ...                                    // parámetros adicionales
   );


bool matrix::Activation(
  matrix&                   matrix_out,    // matriz para obtener los valores
  ENUM_ACTIVATION_FUNCTION  activation     // función de activación
   );


bool matrix::Activation(
  matrix&                   matrix_out,    // matriz para obtener los valores
  ENUM_ACTIVATION_FUNCTION  activation,    // función de activación
  ENUM_MATRIX_AXIS          axis,          // eje
   ...                                    // parámetros adicionales
   );
```

Parámetros

vect_out/matrix_out

[out] Vector o matriz para obtener los valores calculados de la función de activación.

activation

[in] Función de activación de la enumeración ENUM_ACTIVATION_FUNCTION.

axis

[in] Valor de la enumeración ENUM_MATRIX_AXIS (AXIS_HORZ — eje horizontal, AXIS_VERT — eje vertical).

...

[in] Parámetros adicionales necesarios para algunas funciones de activación. Si no se especifica ningún parámetro, se utilizarán los valores por defecto.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Parámetros adicionales

Algunas funciones de activación adoptan parámetros adicionales. Si no se especifica ningún parámetro, se utilizarán los valores por defecto

© 2000-2025, MetaQuotes Ltd.

```
AF_ELU  (Exponential Linear Unit)
  double alpha=1.0

Función de activación: if(x>=0) f(x) = x
                   else f(x) = alpha * (exp(x)-1)


AF_LINEAR
  double alpha=1.0
  double beta=0.0

Función de activación: f(x) = alpha*x + beta


AF_LRELU   (Leaky REctified Linear Unit)
  double alpha=0.3

Función de activación: if(x>=0) f(x)=x
                   else f(x) = alpha*x


AF_RELU  (REctified Linear Unit)
  double alpha=0.0
  double max_value=0.0
  double treshold=0.0

Función de activación: if(alpha==0) f(x) = max(x,0)
                   else if(x>max_value) f(x) = x
                   else f(x) = alpha*(x - treshold)


AF_SWISH
  double beta=1.0

Función de activación: f(x) = x / (1+exp(-x*beta))


AF_TRELU   (Thresholded REctified Linear Unit)
  double theta=1.0

Función de activación: if(x>theta) f(x) = x
                   else f(x) = 0


AF_PRELU   (Parametric REctified Linear Unit)
  double alpha[] - learned array of coeefficients
```

© 2000-2025, MetaQuotes Ltd.

```
Función de activación: if(x[i]>=0) f(x)[i] = x[i]
                       else f(x)[i] = alpha[i] * x[i]
```

Observación

En las redes neuronales artificiales, la función de activación de la neurona define la señal de salida, que viene determinada por la señal de entrada o el conjunto de señales de entrada. La elección de la función de activación influye sustancialmente en las capacidades y el rendimiento de la red neuronal. Las diferentes partes del modelo (capas) pueden utilizar funciones de activación distintas.

Ejemplos de uso de los parámetros adicionales:

```
vector x={0.1, 0.4, 0.9, 2.0, -5.0, 0.0, -0.1};
vector y;

x.Activation(y,AF_ELU);
Print(y);
x.Activation(y,AF_ELU,2.0);
Print(y);

Print("");
x.Activation(y,AF_LINEAR);
Print(y);
x.Activation(y,AF_LINEAR,2.0);
Print(y);
x.Activation(y,AF_LINEAR,2.0,5.0);
Print(y);

Print("");
x.Activation(y,AF_LRELU);
Print(y);
x.Activation(y,AF_LRELU,1.0);
Print(y);
x.Activation(y,AF_LRELU,0.1);
Print(y);

Print("");
x.Activation(y,AF_RELU);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5,1.0);
Print(y);

Print("");
x.Activation(y,AF_SWISH);
Print(y);
```

© 2000-2025, MetaQuotes Ltd.

```
   x.Activation(y,AF_SWISH,2.0);
   Print(y);

   Print("");
   x.Activation(y,AF_TRELU);
   Print(y);
   x.Activation(y,AF_TRELU,0.3);
   Print(y);

   Print("");
   vector a=vector::Full(x.Size(),2.0);
   x.Activation(y,AF_PRELU,a);
   Print(y);

/*  Resultados
   [0.1,0.4,0.9,2,-0.993262053000915,0,-0.095162581964040]
   [0.1,0.4,0.9,2,-1.986524106001829,0,-0.190325163928081]

   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.2,0.8,1.8,4,-10,0,-0.2]
   [5.2,5.8,6.8,9,-5,5,4.8]

   [0.1,0.4,0.9,2,-1.5,0,-0.03]
   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.1,0.4,0.9,2,-0.5,0,-0.01]

   [0.1,0.4,0.9,2,0,0,0]
   [0.2,0.8,0.9,2,-10,0,-0.2]
   [-1.8,-1.2,0.9,2,-12,-2,-2.2]

   [0.052497918747894,0.239475064044981,0.6398545523625035,1.761594155955765,-0.033464
   [0.054983399731247,0.275989792451045,0.7723340415895611,1.964027580075817,-0.000226

   [0,0,0,2,0,0,0]
   [0,0.4,0.9,2,0,0,0]

   [0.1,0.4,0.9,2,-10,0,-0.2]
*/
```

© 2000-2025, MetaQuotes Ltd.

Derivative

Calcula los valores de la derivada de la función de activación y los escribe en el vector/matriz transmitido.

```
bool vector::Derivative(
  vector&                   vect_out,      // vector para obtener los valores
  ENUM_ACTIVATION_FUNCTION  activation,    // función de activación
   ...                                    // parámetros adicionales
   );


bool matrix::Derivative(
  matrix&                   matrix_out,    // matriz para obtener los valores
  ENUM_ACTIVATION_FUNCTION  activation,    // función de activación
   );


bool matrix::Derivative(
  matrix&                   matrix_out,    // matriz para obtener los valores
  ENUM_ACTIVATION_FUNCTION  activation,    // función de activación
  ENUM_MATRIX_AXIS          axis,          // eje
   ...                                    // parámetros adicionales
   );
```

Parámetros

vect_out/matrix_out

[out] Vector o matriz para obtener los valores calculados de la derivada de la función de activación.

activation

[in] Función de activación de la enumeración ENUM_ACTIVATION_FUNCTION.

axis

[in] Valor de la enumeración ENUM_MATRIX_AXIS (AXIS_HORZ — eje horizontal, AXIS_VERT — eje vertical).

...

[in] Los parámetros adicionales son los mismos que en las funciones de activación. Solo algunas funciones de activación adoptan parámetros adicionales. Si no se especifica ningún parámetro, se utilizarán los valores por defecto.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

© 2000-2025, MetaQuotes Ltd.

Las derivadas de las funciones permiten actualizar eficazmente los parámetros del modelo según el error obtenido durante el aprendizaje con la propagación inversa del error.

© 2000-2025, MetaQuotes Ltd.

Loss

Calcula el valor de la función de pérdida.

```
double vector::Loss(
  const vector&       vect_true,     // vector de valores verdaderos
  ENUM_LOSS_FUNCTION  loss,          // función de pérdida
   ...                               // parámetro adicional
   );


double matrix::Loss(
  const matrix&       matrix_true,   // matriz de valores verdaderos
  ENUM_LOSS_FUNCTION  loss,          // función de pérdida
   );


double matrix::Loss(
  const matrix&       matrix_true,   // matriz de valores verdaderos
  ENUM_LOSS_FUNCTION  loss,          // función de pérdida
  ENUM_MATRIX_AXIS    axis,          // eje
   ...                               // parámetro adicional
   );
```

Parámetros

vect_true/matrix_true

[in] Vector o matriz de valores verdaderos.

loss

[in] Función de pérdida de la enumeración ENUM_LOSS_FUNCTION.

axis

[in] Valor de la enumeración ENUM_MATRIX_AXIS (AXIS_HORZ — eje horizontal, AXIS_VERT — eje vertical).

...

[in] Solo la función de pérdida de Hubert (LOSS_HUBER) puede tener el parámetro delta adicional

Valor retornado

Valor double.

Cómo se usa el parámetro delta en la función de pérdida de Huber (LOSS_HUBER)

```
   double delta = 1.0;
   double error = fabs(y - x);
   if(error<delta)
      loss = 0.5 * error^2;
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      loss = 0.5 * delta^2 + delta * (error - delta);
```

Observación

La tarea de entrenar una red neuronal consiste en encontrar coeficientes que minimicen el error en la muestra de entrenamiento, usando para ello la función de pérdida (loss function).

El valor de la función de pérdida describe la magnitud de desviación del valor predicho por el modelo respecto al valor real.

Dependiendo del tipo de tarea a resolver se usarán diferentes funciones de pérdida. Por ejemplo: para un problema de regresión, será mean squared error (MSE), para la clasificación binaria, será binary cross-entropy (BCE)..

Ejemplo de llamada de la función de pérdida de Huber:

```
   vector y_true = {0.0, 1.0, 0.0, 0.0};
   vector y_pred = {0.6, 0.4, 0.4, 0.6};
   double loss=y_pred.Loss(y_true,LOSS_HUBER);
   Print(loss);
   double loss2=y_pred.Loss(y_true,LOSS_HUBER,0.5);
   Print(loss2);

/* Resultado
   0.155
   0.15125
*/
```

© 2000-2025, MetaQuotes Ltd.

LossGradient

Calcula un vector o matriz de gradientes de la función de pérdida.

```
vector vector::LossGradient(
  const vector&       vect_true,     // vector de valores verdaderos
  ENUM_LOSS_FUNCTION  loss,          // tipo de función de pérdida
   ...                               // parámetro adicional
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,   // matriz de valores verdaderos
  ENUM_LOSS_FUNCTION  loss,          // función de pérdida
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,   // matriz de valores verdaderos
  ENUM_LOSS_FUNCTION  loss,          // función de pérdida
  ENUM_MATRIX_AXIS    axis,          // eje
   ...                               // parámetro adicional
   );
```

Parámetros

vect_true/matrix_true

[in] Vector o matriz de valores verdaderos.

loss

[in] Función de pérdida de la enumeración ENUM_LOSS_FUNCTION.

axis

[in] Valor de la enumeración ENUM_MATRIX_AXIS (AXIS_HORZ — eje horizontal, AXIS_VERT — eje vertical).

...

[in] Solo la función de pérdida de Hubert (LOSS_HUBER) puede tener el parámetro delta adicional

Valor retornado

Vector o matriz de valores de los gradientes de la función de pérdida. El gradiente es la derivada parcial de dx (x es el valor previsto) de la función de pérdida en un punto determinado.

Observación

Los gradientes se utilizan en las redes neuronales para ajustar los valores de la matriz de pesos cuando el error se propaga de forma inversa durante el entrenamiento del modelo.

© 2000-2025, MetaQuotes Ltd.

La tarea de entrenar una red neuronal consiste en encontrar coeficientes que minimicen el error en la muestra de entrenamiento, usando para ello la función de pérdida (loss function).

Dependiendo del tipo de tarea a resolver se usarán diferentes funciones de pérdida. Por ejemplo: para un problema de regresión, será mean squared error (MSE), para la clasificación binaria, será binary cross-entropy (BCE)..

Ejemplo de cálculo de gradientes de la función de pérdida

```
   matrixf y_true={{ 1, 2, 3, 4 },
                   { 5, 6, 7, 8 },
                   { 9,10,11,12 }};
   matrixf y_pred={{ 1, 2, 3, 4 },
                   {11,10, 9, 8 },
                   { 5, 6, 7,12 }};
   matrixf loss_gradient =y_pred.LossGradient(y_true,LOSS_MAE);
   matrixf loss_gradienth=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_HORZ);
   matrixf loss_gradientv=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_VERT);
   Print("loss gradients\n",loss_gradient);
   Print("loss gradients on horizontal axis\n",loss_gradienth);
   Print("loss gradients on vertical axis\n",loss_gradientv);

/* Resultado
   loss gradients
   [[0,0,0,0]
    [0.083333336,0.083333336,0.083333336,0]
    [-0.083333336,-0.083333336,-0.083333336,0]]
   loss gradients on horizontal axis
   [[0,0,0,0]
    [0.33333334,0.33333334,0.33333334,0]
    [-0.33333334,-0.33333334,-0.33333334,0]]
   loss gradients on vertical axis
   [[0,0,0,0]
    [0.25,0.25,0.25,0]
    [-0.25,-0.25,-0.25,0]]
*/
```

© 2000-2025, MetaQuotes Ltd.

RegressionMetric

Calcula la métrica de regresión para evaluar la calidad de los datos previstos respecto a los datos reales

```
double vector::RegressionMetric(
   const vector&           vector_true,   // vector de valores verdaderos
   ENUM_REGRESSION_METRIC  metric         // tipo de métrica
   );

double matrix::RegressionMetric(
   const matrix&           matrix_true,   // matriz de valores verdaderos
   ENUM_REGRESSION_METRIC  metric         // tipo de métrica
);

vector matrix::RegressionMetric(
   const matrix&           matrix_true,   // matriz de valores verdaderos
   ENUM_REGRESSION_METRIC  metric,        // tipo de métrica
   int                     axis           // eje
   );
```

Parámetros

vector_true/matrix_true

[in] Vector o matriz de valores verdaderos.

metric

[in] Tipo de métrica de la enumeración ENUM_REGRESSION_METRIC.

axis

[in] Eje. 0 — eje horizontal, 1 — eje vertical.

Valor retornado

La métrica calculada supone una valoración de la calidad de los datos predichos respecto a los datos verdaderos.

Observación

- REGRESSION_MAE — media de las diferencias absolutas entre los valores previstos y los valores reales correspondientes
- REGRESSION_MSE — media de los cuadrados de las diferencias entre los valores previstos y los valores reales correspondientes
- REGRESSION_RMSE — raíz cuadrada de MSE
- REGRESSION_R2 - 1 — MSE(regresión) / MSE(media)
- REGRESSION_MAPE — error absoluto medio (MAE) en porcentaje
- REGRESSION_MSPE — error cuadrático medio (MSE) en porcentaje
- REGRESSION_RMSLE — RMSE calculado en escala logarítmica

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
   vector y_true = {3, -0.5, 2, 7};
   vector y_pred = {2.5, 0.0, 2, 8};
//---
   double mse=y_pred.RegressionMetric(y_true,REGRESSION_MSE);
   Print("mse=",mse);
//---
   double mae=y_pred.RegressionMetric(y_true,REGRESSION_MAE);
   Print("mae=",mae);
//---
   double r2=y_pred.RegressionMetric(y_true,REGRESSION_R2);
   Print("r2=",r2);

  /* Resultado
   mae=0.375
   mse=0.5
   r2=0.9486081370449679
   */
```

© 2000-2025, MetaQuotes Ltd.

ConfusionMatrix

Calcula la matriz de error. El método se aplica a un vector de valores predichos.

```
matrix vector::ConfusionMatrix(
   const vector&        vect_true      // vector de valores verdaderos
   );


matrix vector::ConfusionMatrix(
   const vector&        vect_true,     // vector de valores verdaderos
   uint                 label          // valor de la etiqueta
   );
```

Parámetros

vect_true

[in] Vector de valores verdaderos.

эпохи

[in] Valor de la etiqueta para calcular la matriz de error.

Valor retornado

Matriz de errores. Si no se especifica ningún valor de etiqueta, se retornará una matriz de error multiclase en la que cada etiqueta se emparejará con todas las demás individualmente. Si se especifica un valor de etiqueta, se retornará una matriz de 2 x 2, donde la etiqueta especificada se considerará positiva y todas las demás etiquetas serán negativas (ovr, one vs rest).

Observación

La matriz de error C es tal que Cij es igual al número de observaciones que se encuentran en el grupo i y también, según los pronósticos, se encuentran en el grupo j. Así, en la clasificación binaria, el número de verdaderos negativos (NT) será C00, el de falsos negativos (FN) será C10, el de verdaderos positivos (TP) será C11 y el de falsos positivos (FP) será C01.

Es decir, esta matriz puede representarse gráficamente de la forma siguiente:

TN	FP
FN	TP

Los tamaños del vector de valores verdaderos y del vector de valores predichos deberán ser iguales.

Ejemplo:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
```

© 2000-2025, MetaQuotes Ltd.

```
   matrix confusion=y_pred.ConfusionMatrix(y_true);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,0);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,1);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,2);
   Print(confusion);


/*
  [[3,0,0,0,0,0,0,0,0,0]
   [0,3,0,0,0,0,0,0,0,0]
   [0,0,1,0,1,0,0,1,0,0]
   [0,0,0,1,0,0,0,1,0,0]
   [0,0,1,0,3,0,0,0,0,1]
   [0,0,0,0,0,2,0,0,0,0]
   [1,0,0,0,0,1,1,0,0,0]
   [0,0,0,0,0,0,0,2,0,1]
   [0,0,1,0,0,0,0,0,0,1]
   [0,0,0,0,0,0,0,0,0,4]]
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
   [2,1]]
*/
```

© 2000-2025, MetaQuotes Ltd.

ConfusionMatrixMultilabel

Calcula la matriz de error para cada etiqueta. El método se aplica a un vector de valores predichos.

```
uint vector::ConfusionMatrixMultiLabel(
   const vector&       vect_true,     // vector de valores verdaderos
   matrix&             confusions[]   // array de matrices de error calculadas
   );
```

Parámetros

vect_true

[in] Vector de valores verdaderos.

confusions

[out] Array de matrices 2 x 2 con las matrices de error calculadas de cada etiqueta.

Valor retornado

Tamaño del array de matrices de error calculadas. En caso de fallo, se retornará 0

Observación

El array resultante puede ser dinámico o estático. Si el array es estático, deberá tener un tamaño no inferior al número de clases.

Los tamaños del vector de valores verdaderos y del vector de valores predichos deberán ser iguales.

Ejemplo:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
   matrix label_confusions[12];

   uint   res=y_pred.ConfusionMatrixMultiLabel(y_true,label_confusions);
   Print("res=",res,"  size=",label_confusions.Size());
   for(uint i=0; i<res; i++)
      Print(label_confusions[i]);


/*
  res=10  size=12
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
   [2,1]]
```

© 2000-2025, MetaQuotes Ltd.

```
   [[28,0]
    [1,1]]
   [[24,1]
    [2,3]]
   [[27,1]
    [0,2]]
   [[27,0]
    [2,1]]
   [[25,2]
    [1,2]]
   [[28,0]
    [2,0]]
   [[23,3]
    [0,4]]
*/
```

© 2000-2025, MetaQuotes Ltd.

ClassificationMetric

Calcula la métrica clasificatoria para evaluar la calidad de los datos previstos respecto a los datos reales. El método se aplica a un vector de valores predichos.

```
vector vector::ClassificationMetric(
   const vector&               vect_true,     // vector de valores verdaderos
   ENUM_CLASSIFICATION_METRIC metric         // tipo de métrica
   );


vector vector::ClassificationMetric(
   const vector&               vect_true,     // vector de valores verdaderos
   ENUM_CLASSIFICATION_METRIC metric         // tipo de métrica
   ENUM_AVERAGE_MODE           mode           // modo de promediación
   );
```

Parámetros

vect_true

[in] Vector de valores verdaderos.

metric

[in] Tipo de métrica de la enumeración ENUM_CLASSIFICATION_METRIC. Se usan los valores salvo CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION y CLASSIFICATION_ROC_AUC (que se usan en el método ClassificationScore).

mode

[in] Modo de promediación de la enumeración ENUM_AVERAGE_MODE. Se usa para las métricas CLASSIFICATION_F1, CLASSIFICATION_JACCARD, CLASSIFICATION_PRECISION y CLASSIFICATION_RECALL.

Valor retornado

Vector que contiene la métrica calculada. En el caso del modo de promediado AVERAGE_NONE, el vector contendrá valores métricos para cada clase sin promediar. (Por ejemplo, para la clasificación binaria, esto serían 2 métricas para false y true, respectivamente).

Observación sobre los modos de promediación

AVERAGE_BINARY solo tiene sentido para la clasificación binaria.

AVERAGE_MICRO: calcula métricas de forma global contando el número total de verdaderos, falsos negativos y falsos positivos.

AVERAGE_MACRO: calcula métricas para cada etiqueta y encuentra su promedio no ponderado. Esto no considera el desequilibrio de las etiquetas.

AVERAGE_WEIGHTED: calcula las métricas para cada etiqueta y encuentra su promedio ponderado según el soporte (el número de ejemplares reales para cada etiqueta). Este cálculo cambia la macro para considerar el desequilibrio de las etiquetas, lo cual puede dar como resultado una medida F que no se encuentre entre la precisión y la recuperación.

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

vector accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
Print("accuracy=",accuracy);
vector balanced=y_pred.ClassificationMetric(y_true,CLASSIFICATION_BALANCED_ACCURACY
Print("balanced=",balanced);
Print("");

vector f1_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MICRO)
Print("f1_micro=",f1_micro);
vector f1_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MACRO)
Print("f1_macro=",f1_macro);
vector f1_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_WEI
Print("f1_weighted=",f1_weighted);
vector f1_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_NONE);
Print("f1_none=",f1_none);
Print("");

vector jaccard_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVER
Print("jaccard_micro=",jaccard_micro);
vector jaccard_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVER
Print("jaccard_macro=",jaccard_macro);
vector jaccard_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,A
Print("jaccard_weighted=",jaccard_weighted);
vector jaccard_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVERA
Print("jaccard_none=",jaccard_none);
Print("");

vector precision_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,
Print("precision_micro=",precision_micro);
vector precision_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,
Print("precision_macro=",precision_macro);
vector precision_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISI
Print("precision_weighted=",precision_weighted);
vector precision_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,A
Print("precision_none=",precision_none);
Print("");

vector recall_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAG
Print("recall_micro=",recall_micro);
vector recall_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAG
Print("recall_macro=",recall_macro);
vector recall_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVE
Print("recall_weighted=",recall_weighted);
```

© 2000-2025, MetaQuotes Ltd.

```
   vector recall_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAGE
   Print("recall_none=",recall_none);
   Print("");

//--- binary classification
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};

   vector f1_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_F1,AVERAGE_
   Print("f1_bin=",f1_bin);
   vector jaccard_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_JACCAR
   Print("jaccard_bin=",jaccard_bin);
   vector precision_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_PREC
   Print("precision_bin=",precision_bin);
   vector recall_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_RECALL,
   Print("recall_bin=",recall_bin);


/*
  accuracy=[0.6666666666666666]
  balanced=[0.6433333333333333]

  f1_micro=[0.6666666666666666]
  f1_macro=[0.6122510822510823]
  f1_weighted=[0.632049062049062]
  f1_none=[0.8571428571428571,1,0.3333333333333333,0.6666666666666666,0.6666666666666

  jaccard_micro=[0.5]
  jaccard_macro=[0.4921428571428572]
  jaccard_weighted=[0.5056349206349205]
  jaccard_none=[0.75,1,0.2,0.5,0.5,0.6666666666666666,0.3333333333333333,0.4,0,0.5714

  precision_micro=[0.6666666666666666]
  precision_macro=[0.6571428571428571]
  precision_weighted=[0.6706349206349207]
  precision_none=[0.75,1,0.3333333333333333,1,0.75,0.6666666666666666,1,0.5,0,0.571428

  recall_micro=[0.6666666666666666]
  recall_macro=[0.6433333333333333]
  recall_weighted=[0.6666666666666666]
  recall_none=[1,1,0.3333333333333333,0.5,0.6,1,0.3333333333333333,0.6666666666666666,

  f1_bin=[0.4444444444444445]
  jaccard_bin=[0.2857142857142857]
  precision_bin=[0.5]
  recall_bin=[0.4]
*/
```

© 2000-2025, MetaQuotes Ltd.

ClassificationScore

Calcula la métrica clasificatoria para evaluar la calidad de los datos previstos respecto a los datos reales.

A diferencia de otros métodos de la sección "Aprendizaje automático", este método se aplica a un vector de valores verdaderos en lugar de a un vector de valores predichos.

```
vector vector::ClassificationScore(
   const matrix&               pred_scores,   // matriz que contiene la distribución de
   ENUM_CLASSIFICATION_METRIC metric         // tipo de métrica
   ENUM_AVERAGE_MODE           mode          // modo de promediación
   );


vector vector::ClassificationScore(
   const matrix&               pred_scores,   // matriz que contiene la distribución de
   ENUM_CLASSIFICATION_METRIC metric         // tipo de métrica
   int                         param         // parámetro adicional
   );
```

Parámetros

pred_scores

[in] Matriz que contiene un conjunto de vectores horizontales con las probabilidades para cada clase. El número de filas de la matriz deberá corresponderse con el tamaño del vector de valores verdaderos.

metric

[in] Tipo de métrica de la enumeración ENUM_CLASSIFICATION_METRIC. Se usan los valores CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION и CLASSIFICATION_ROC_AUC.

mode

[in] Modo de promediación de la enumeración ENUM_AVERAGE_MODE. Se usa para las métricas CLASSIFICATION_AVERAGE_PRECISION y CLASSIFICATION_ROC_AUC.

param

[in] Para la métrica CLASSIFICATION_TOP_TOP_K_ACCURACY, se debe especificar un valor K entero en lugar del modo de promediación.

Valor retornado

Vector que contiene la métrica calculada. En el caso del modo de promediado AVERAGE_NONE, el vector contendrá valores métricos para cada clase sin promediar. (Por ejemplo, para la clasificación binaria, esto serían 2 métricas para false y true, respectivamente).

Observación sobre los modos de promediación

© 2000-2025, MetaQuotes Ltd.

AVERAGE_BINARY solo tiene sentido para la clasificación binaria.

AVERAGE_MICRO — calcular las métricas de forma global, considerando cada elemento de la matriz de indicadores de etiquetas como una etiqueta. Por matriz de indicadores de etiquetas entendemos una matriz con un conjunto de probabilidades para cada etiqueta.

AVERAGE_MACRO: calcula métricas para cada etiqueta y encuentra su promedio no ponderado. Esto no considera el desequilibrio de las etiquetas.

AVERAGE_WEIGHTED: calcula las métricas para cada etiqueta y encuentra su promedio ponderado según el soporte (el número de ejemplares reales para cada etiqueta).

Observación

Para la clasificación binaria, no solo puede introducirse una matriz n x 2, en la que la primera columna contiene las probabilidades de la etiqueta negativa y la segunda las de la etiqueta positiva, sino también una matriz formada por una columna con probabilidades positivas. Esto se debe a que los modelos de clasificación binaria pueden retornar tanto dos probabilidades como una probabilidad para una etiqueta positiva.

Ejemplo:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   //vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

//--- label scores          0         1         2         3         4         5
   matrix y_scores={{0.000109, 0.000186, 0.000449, 0.000052, 0.000002, 0.000022, 0.000
                    {0.000091, 0.081956, 0.916816, 0.001106, 0.000006, 0.000002, 0.000
                    {0.000108, 0.972863, 0.003600, 0.000021, 0.010479, 0.000015, 0.000
                    {0.925425, 0.000080, 0.002913, 0.000057, 0.000274, 0.000638, 0.063
                    {0.000060, 0.000126, 0.000006, 0.000000, 0.993513, 0.000000, 0.000
                    {0.000016, 0.982124, 0.000045, 0.000002, 0.008445, 0.000001, 0.000
                    {0.000000, 0.000040, 0.000001, 0.000000, 0.989395, 0.000167, 0.000
                    {0.000795, 0.002938, 0.023447, 0.007418, 0.021838, 0.002476, 0.000
                    {0.000091, 0.000226, 0.000038, 0.000007, 0.000048, 0.854910, 0.068
                    {0.000000, 0.000000, 0.000000, 0.000000, 0.003004, 0.000000, 0.000
                    {0.998856, 0.000009, 0.000976, 0.000002, 0.000000, 0.000013, 0.000
                    {0.000178, 0.000446, 0.000326, 0.000033, 0.000193, 0.000071, 0.998
                    {0.000005, 0.000016, 0.000153, 0.000045, 0.004110, 0.000012, 0.000
                    {0.994188, 0.000003, 0.002584, 0.000005, 0.000005, 0.000100, 0.000
                    {0.000173, 0.990569, 0.000792, 0.000040, 0.001798, 0.000035, 0.000
                    {0.000000, 0.000537, 0.000008, 0.005080, 0.000046, 0.992910, 0.000
                    {0.000127, 0.000003, 0.000003, 0.000000, 0.001583, 0.000000, 0.000
                    {0.000001, 0.000012, 0.000072, 0.000020, 0.000000, 0.000000, 0.000
                    {0.000020, 0.000105, 0.001139, 0.901343, 0.002132, 0.083873, 0.000
                    {0.000002, 0.000048, 0.000019, 0.000000, 0.999347, 0.000002, 0.000
                    {0.000059, 0.001344, 0.612502, 0.002749, 0.000229, 0.000678, 0.000
                    {0.000586, 0.000740, 0.001625, 0.000007, 0.269341, 0.000076, 0.016
                    {0.009547, 0.018055, 0.283795, 0.071079, 0.426074, 0.082335, 0.036
                    {0.002506, 0.002545, 0.001148, 0.005659, 0.020416, 0.000112, 0.006
```

© 2000-2025, MetaQuotes Ltd.

```
                     {0.001263, 0.001769, 0.000293, 0.000011, 0.000302, 0.881768, 0.112
                     {0.002904, 0.002909, 0.013421, 0.001461, 0.007519, 0.001251, 0.000
                     {0.000055, 0.001080, 0.893158, 0.000000, 0.104492, 0.000159, 0.001
                     {0.000344, 0.002693, 0.071184, 0.000262, 0.000001, 0.000003, 0.000
                     {0.001404, 0.009375, 0.002638, 0.229189, 0.000064, 0.000896, 0.007
                     {0.491140, 0.000125, 0.000024, 0.000302, 0.000038, 0.034947, 0.473

   vector top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,1);
   Print("top 1 accuracy score = ",top_k);
   top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top 2 accuracy score = ",top_k);
   vector y_true2={0, 1, 2, 2};
   matrix y_score2={{0.5, 0.2, 0.2},  // 0 is in top 2
                    {0.3, 0.4, 0.2},  // 1 is in top 2
                    {0.2, 0.4, 0.3},  // 2 is in top 2
                    {0.7, 0.2, 0.1}}; // 2 isn't in top 2
   top_k=y_true2.ClassificationScore(y_score2,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top k = ",top_k);
   Print("");

   vector ap_micro=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISIO
   Print("average precision score micro = ",ap_micro);
   vector ap_macro=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISIO
   Print("average precision score macro = ",ap_macro);
   vector ap_weighted=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PREC
   Print("average precision score weighted = ",ap_weighted);
   vector ap_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISION
   Print("average precision score none = ",ap_none);
   Print("");

   vector area_micro=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAG
   Print("roc auc score micro = ",area_micro);
   vector area_macro=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAG
   Print("roc auc score macro = ",area_macro);
   vector area_weighted=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score weighted = ",area_weighted);
   vector area_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAGE
   Print("roc auc score none = ",area_none);
   Print("");

//--- binary classification
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};
   vector y_score_true={0.3,0.7,0.1,0.6,0.9,0.0,0.4,0.2,0.8};
   matrix y_score1_bin(y_score_true.Size(),1);
   y_score1_bin.Col(y_score_true,0);
   matrix y_scores_bin={{0.7, 0.3},
                        {0.3, 0.7},
                        {0.9, 0.1},
```

© 2000-2025, MetaQuotes Ltd.

```
                        {0.4, 0.6},
                        {0.1, 0.9},
                        {1.0, 0.0},
                        {0.6, 0.4},
                        {0.8, 0.2},
                        {0.2, 0.8}};

   vector ap=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECIS
   Print("average precision score binary = ",ap);
   vector ap2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_AVERAGE_PRECI
   Print("average precision score binary = ",ap2);
   vector ap3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECI
   Print("average precision score none = ",ap3);
   Print("");

   vector area=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVER
   Print("roc auc score binary = ",area);
   vector area2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score binary = ",area2);
   vector area3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score none = ",area3);


/*
  top 1 accuracy score = [0.6666666666666666]
  top 2 accuracy score = [1]
  top k = [0.75]

  average precision score micro = [0.8513333333333333]
  average precision score macro = [0.9326666666666666]
  average precision score weighted = [0.9333333333333333]
  average precision score none = [1,1,0.7,1,0.9266666666666666,0.8333333333333333,1,0.

  roc auc score micro = [0.9839506172839506]
  roc auc score macro = [0.9892068783068803]
  roc auc score weighted = [0.9887354497354497]
  roc auc score none = [1,1,0.9506172839506173,1,0.984,0.9821428571428571,1,0.9753086

  average precision score binary = [0.7961904761904761]
  average precision score binary = [0.7961904761904761]
  average precision score none = [0.7678571428571428,0.7961904761904761]

  roc auc score binary = [0.7]
  roc auc score binary = [0.7]
  roc auc score none = [0.7,0.7]
*/
```

© 2000-2025, MetaQuotes Ltd.

PrecisionRecall

Calcula los valores para construir la curva precision-recall. Este método, al igual que el método ClassificationScore, se aplica al vector de valores verdaderos.

```
bool vector::PrecisionRecall(
   const matrix&                 pred_scores,   // matriz que contiene la distribución
   const ENUM_ENUM_AVERAGE_MODE  mode           // modo de promediación
   matrix&                       precision,     // valores precision calculados para c
   matrix&                       recall,        // valores recall calculados para cada
   matrix&                       thresholds,    // valores umbral clasificados en orde
   );
```

Parámetros

pred_scores

[in] Matriz que contiene un conjunto de vectores horizontales con probabilidades para cada clase. El número de filas de la matriz debe corresponderse con el tamaño del vector de valores verdaderos.

mode

[in] Modo de promediación de la enumeración ENUM_AVERAGE_MODE. Se usan solo AVERAGE_NONE, AVERAGE_BINARY y AVERAGE_MICRO.

precision

[out] Matriz con los valores calculados de la curva precision. Si no hay promediación (AVERAGE_NONE), el número de filas de la matriz se corresponderá con el número de clases del modelo. El número de columnas se corresponde con el tamaño del vector de valores verdaderos (o al número de filas de la matriz de distribución de probabilidad pred_score). En caso de micro-promediación, el número de filas de la matriz se corresponderá con el número total de valores umbral excluyendo los dobles.

recall

[out] Matriz con los valores calculados de la curva recall.

threshold

[out] Matriz de valores umbral obtenida clasificando la matriz de probabilidades

Observación

Consulte las notas para el método ClassificationScore.

Ejemplo

Ejemplo de recopilación de estadísticas del modelo mnist.onnx (precisión 99%).

```
//--- data for classification metrics
   vectorf y_true(images);
   vectorf y_pred(images);
   matrixf y_scores(images,10);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- input-output
   matrixf image(28,28);
   vectorf result(10);

//--- testing
   for(int test=0; test<images; test++)
     {
      image=test_data[test].image;
      if(!OnnxRun(model,ONNX_DEFAULT,image,result))
        {
         Print("OnnxRun error ",GetLastError());
         break;
        }
      result.Activation(result,AF_SOFTMAX);
      //--- collect data
      y_true[test]=(float)test_data[test].label;
      y_pred[test]=(float)result.ArgMax();
      y_scores.Row(result,test);
     }    }
```

Cálculo de la precisión

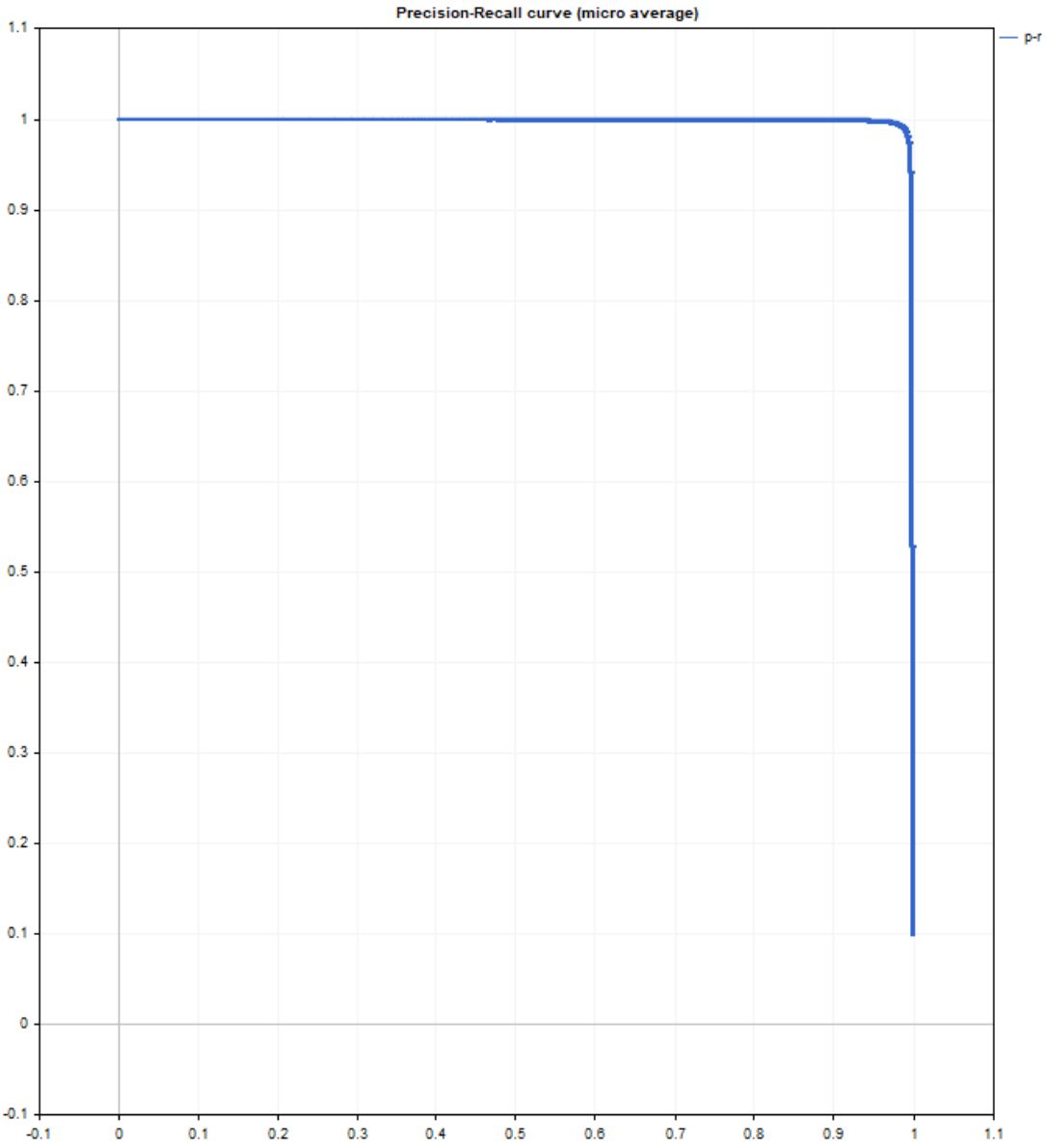
```
   vectorf accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
   PrintFormat("accuracy=%f",accuracy[0]);

accuracy=0.989000
```

Ejemplo de gráfico precision-recall en el que los valores precision se representan en el eje y, mientras que los valores recall se representan en el eje x. También se obtienen gráficos precision y recall independientes, en los que los valores umbral se representan en el eje x.

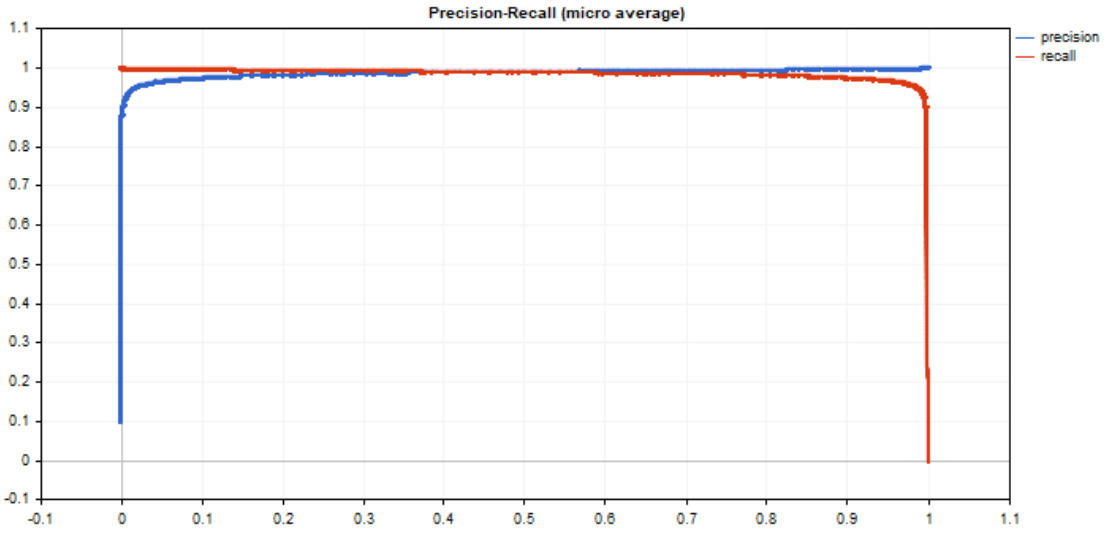
```
   if(y_true.PrecisionRecall(y_scores,AVERAGE_MICRO,mat_precision,mat_recall,mat_thres
     {
      double precision[],recall[],thres[];
      ArrayResize(precision,mat_thres.Cols());
      ArrayResize(recall,mat_thres.Cols());
      ArrayResize(thres,mat_thres.Cols());

      for(uint i=0; i<thres.Size(); i++)
        {
         precision[i]=mat_precision[0][i];
         recall[i]=mat_recall[0][i];
         thres[i]=mat_thres[0][i];
        }
      thres[0]=thres[1]+0.001;

      PlotCurve("Precision-Recall curve (micro average)","p-r","",recall,precision);
      Plot2Curves("Precision-Recall (micro average)","precision","recall",thres,precis
     }
```

© 2000-2025, MetaQuotes Ltd.

Resultado:



© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

ReceiverOperatingCharacteristic

Calcula los valores para construir la curva Receiver Operating Characteristic (ROC). Este método, al igual que el método ClassificationScore, se aplica al vector de valores verdaderos.

```
bool vector::ReceiverOperatingCharacteristic(
   const matrix&                 pred_scores,   // matriz que contiene la distribución
   const ENUM_ENUM_AVERAGE_MODE  mode           // modo de promediación
   matrix&                       fpr,           // valores false positive rate calcula
   matrix&                       tpr,           // valores true positive rate calculad
   matrix&                       thresholds,    // valores umbral clasificados en orde
   );
```

Parámetros

pred_scores

[in] Matriz que contiene un conjunto de vectores horizontales con probabilidades para cada clase. El número de filas de la matriz debe corresponderse con el tamaño del vector de valores verdaderos.

mode

[in] Modo de promediación de la enumeración ENUM_AVERAGE_MODE. Se usan solo AVERAGE_NONE, AVERAGE_BINARY y AVERAGE_MICRO.

fpr

[out] Matriz con los valores calculados de la curva false positive rate. Si no hay promediación (AVERAGE_NONE), el número de filas de la matriz se corresponderá con el número de clases del modelo. El número de columnas se corresponde con el tamaño del vector de valores verdaderos (o al número de filas de la matriz de distribución de probabilidad pred_score). En caso de micro-promediación, el número de filas de la matriz se corresponderá con el número total de valores umbral excluyendo los dobles.

tpr

[out] Matriz con los valores calculados de la curva true positive rate.

threshold

[out] Matriz de valores umbral obtenida clasificando la matriz de probabilidades

Observación

Consulte las notas para el método ClassificationScore.

Ejemplo

Ejemplo de muestra de los gráficos ROC, donde los valores tpr se representan en el eje y, mientras que los valores fpr se representan en el eje x. También se obtienen gráficos fpr y tpr independientes, en los que los valores umbral se representan en el eje x.

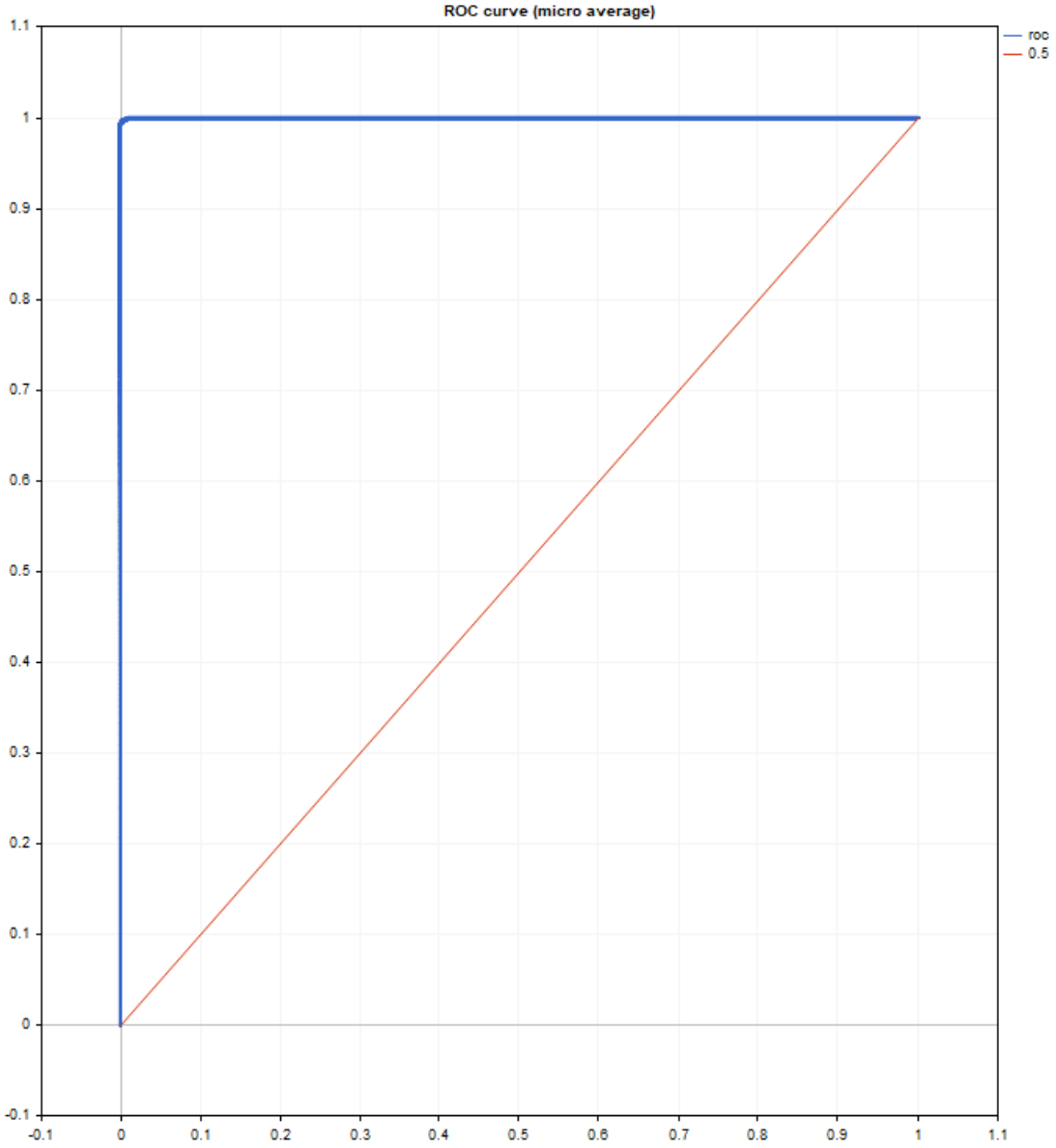
```
   matrixf mat_thres;
   matrixf mat_fpr;
```

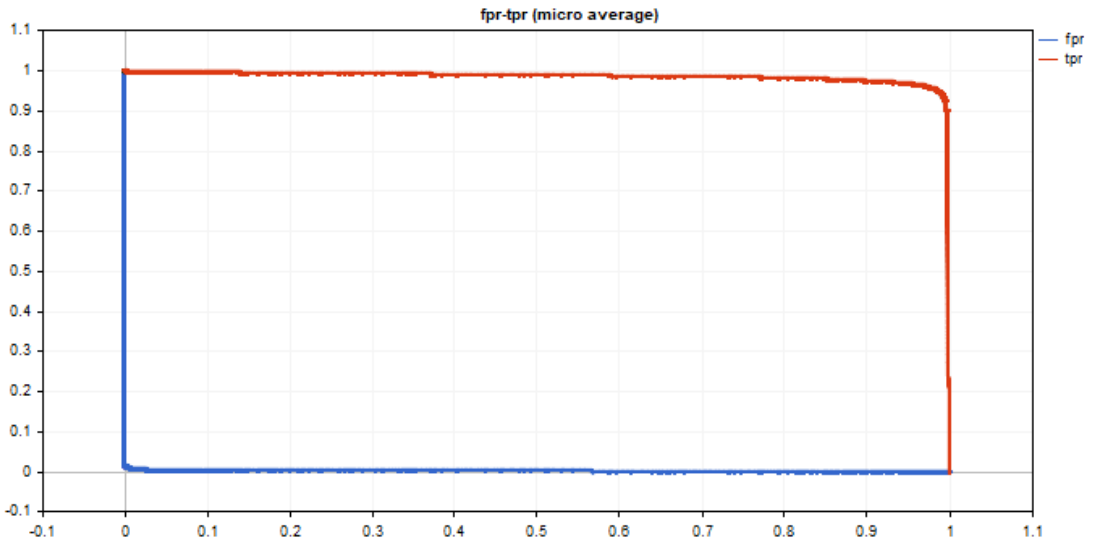
© 2000-2025, MetaQuotes Ltd.

```
   matrixf mat_tpr;

   if(y_true.ReceiverOperatingCharacteristic(y_scores,AVERAGE_MICRO,mat_fpr,mat_tpr,ma
     {
      double fpr[],tpr[],thres[];
      ArrayResize(fpr,mat_thres.Cols());
      ArrayResize(tpr,mat_thres.Cols());
      ArrayResize(thres,mat_thres.Cols());

      for(uint i=0; i<fpr.Size(); i++)
        {
         fpr[i]=mat_fpr[0][i];
         tpr[i]=mat_tpr[0][i];
         thres[i]=mat_thres[0][i];
        }
      thres[0]=thres[1]+0.001;

      PlotCurve("ROC curve (micro average)","roc","0.5",fpr,tpr);
      Plot2Curves("fpr-tpr (micro average)","fpr","tpr",thres,fpr,tpr);
     }
```

Resultado:

© 2000-2025, MetaQuotes Ltd.





© 2000-2025, MetaQuotes Ltd.

El código de muestra de gráficos es elemental y se basa en la biblioteca estándar <Graphics/Graphic.mqh>.

Se han utilizado los datos de prueba del modelo mnist.onnx, cuyo código se presenta en la descripción del método PrecisionRecall.

ROC AUC está próximo al ideal.

```
roc auc score micro = [0.99991]
```

© 2000-2025, MetaQuotes Ltd.

OpenBLAS Methods

OpenBLAS is a high-performance open-source linear algebra library that implements BLAS (Basic Linear Algebra Subprograms) and some LAPACK functions. OpenBLAS is designed to improve computational performance, particularly in matrix and vector operations, which are often used in scientific and engineering tasks such as machine learning, numerical methods, and simulations.

Key features of OpenBLAS:

- Multithreading support: OpenBLAS can efficiently use multiple processor cores for parallel computations, significantly accelerating operations on multiprocessor systems.
- Optimization for processor architectures: OpenBLAS includes optimized builds for various processors such as Intel, AMD, ARM and others. The library automatically detects processor characteristics and selects the most suitable function implementations.
- Extensive BLAS operation support: OpenBLAS implements core BLAS functions, including vector operations (e.g., vector addition and dot product), matrix operations (multiplication), and vector-matrix operations.
- LAPACK compatibility: The library supports LAPACK (Linear Algebra PACKage) functions for more complex linear algebra operations, such as solving systems of linear equations, calculating matrix eigenvalues, and others.
- High performance: Compared to other BLAS libraries, OpenBLAS often shows better results due to hand-optimization for specific processor architectures.

Applications

OpenBLAS is widely used in applications involving numerical computations:

- Training neural networks and other machine learning tasks.
- Scientific computing (e.g. modeling of physical processes).
- Processing and analyzing large amounts of data.

The library is integrated into many popular scientific software packages such as NumPy, SciPy, and TensorFlow, which rely on high-performance linear algebra operations.

OpenBLAS is an excellent choice for those seeking an open-source solution for high-performance computing, particularly when working with large matrices and vectors.

Function	Action
SingularValueDecompositionDC	Singular Value Decomposition, "divide-and-conquer" algorithm. This algorithm is considered the fastest among other SVD algorithms (lapack function GESDD).
SingularValueDecompositionQR	Singular Value Decomposition, QR algorithm. This algorithm is considered a classical SVD algorithm (lapack function GESVD).

© 2000-2025, MetaQuotes Ltd.

Function	Action
SingularValueDecompositionQRPivot	Singular Value Decomposition, QR with pivoting algorithm (lapack function GESVDQ).
SingularValueDecompositionBisect	Singular Value Decomposition, bisection algorithm (lapack function GESVDX).
SingularValueDecompositionJacobiHigh	Singular Value Decomposition, Jacobi high level algorithm (lapack function GEJSV).
SingularValueDecompositionJacobiLow	Singular Value Decomposition, Jacobi low level algorithm (lapack function GESVJ). The method computes small singular values and their singular vectors with much greater accuracy than other SVD routines in certain cases.
SingularValueDecompositionBidiagDC	Singular Value Decomposition, divide-and-conquer algorithm for bidiagonal matrices (lapack function BDSVDX).
SingularValueDecompositionBidiagBisect	Singular Value Decomposition, bisection algorithm for bidiagonal matrices (lapack function BDSVDX).
EigenSolver	Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (lapack function GEEV).
EigenSolver2	Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (lapack function GGEV).
EigenSolverX	Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GEEVX).
EigenSolverShur	Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (lapack function GEES). See also Schur decomposition.
EigenSymmetricDC	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the divide-and-conquer algorithm (lapack functions SYEVD, HEEVD).
EigenSymmetricQR	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex

© 2000-2025, MetaQuotes Ltd.

Function	Action
	conjugate) matrix using the QR algorithm (lapack functions SYEV, HEEV).
EigenSymmetricRobust	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (lapack functions SYEVR, HEEVR).
EigenSymmetricBisect	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the bisection algorithm (lapack functions SYEVX, HEEVX).
SingularSpectrumAnalysisSpectrum	A method function for calculating the relative contributions of spectral components based on their eigenvalues.
SingularSpectrumAnalysisForecast	A method function for calculating reconstructed and predicted data using spectral components of the input time series.
SingularSpectrumAnalysisReconstructComponents	A method function for calculating reconstructed components of the input time series and their contributions.
SingularSpectrumAnalysisReconstructSeries	A method function for calculating the reconstructed time series using the first component_count components.

© 2000-2025, MetaQuotes Ltd.

Singular value decomposition

This section features functions for decomposing a matrix into three components: orthogonal matrices and a diagonal matrix of singular values. SVD is applied to solve various linear algebra problems such as data dimensionality reduction, image compression, solving systems of equations, and data analysis and optimization. The main functions allow you to compute singular values and vectors, reconstruct matrices, and approximate matrices with reduced rank accuracy.

Function	Action
SingularValueDecompositionDC	Singular Value Decomposition, "divide-and-conquer" algorithm. This algorithm is considered the fastest among other SVD algorithms (lapack function GESDD).
SingularValueDecompositionQR	Singular Value Decomposition, QR algorithm. This algorithm is considered a classical SVD algorithm (lapack function GESVD).
SingularValueDecompositionQRPivot	Singular Value Decomposition, QR with pivoting algorithm (lapack function GESVDQ).
SingularValueDecompositionBisect	Singular Value Decomposition, bisection algorithm (lapack function GESVDX).
SingularValueDecompositionJacobiHigh	Singular Value Decomposition, Jacobi high level algorithm (lapack function GEJSV).
SingularValueDecompositionJacobiLow	Singular Value Decomposition, Jacobi low level algorithm (lapack function GESVJ). The method computes small singular values and their singular vectors with much greater accuracy than other SVD routines in certain cases.
SingularValueDecompositionBidiagDC	Singular Value Decomposition, divide-and-conquer algorithm for bidiagonal matrices (lapack function BDSVDX).
SingularValueDecompositionBidiagBisect	Singular Value Decomposition, bisection algorithm for bidiagonal matrices (lapack function BDSVDX).
SingularValueDecompositionBidiagQR	Computes the singular value decomposition of a general matrix that has been reduced to bidiagonal form by the method ReduceToBidiagonal. Lapack function BDSQR.

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionDC

Singular Value Decomposition, "divide-and-conquer" algorithm. This algorithm is considered the fastest among other SVD algorithms (lapack function GESDD).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z      jobz,     // how to computed
   vector&         S,        // vector of computed singular values
   matrix&         U,        // matrix of computed left vectors U
   matrix&         VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z      jobz,     // how to computed
   vectorf&        S,        // vector of computed singular values
   matrixf&        U,        // matrix of computed left vectors U
   matrixf&        VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z      jobz,     // how to computed
   vector&         S,        // vector of computed singular values
   matrixc&        U,        // matrix of computed left vectors U
   matrixc&        VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z      jobz,                  // how to computed
   vectorf&        singular_values,       // vector of computed singular values
   matrixcf&       U,                     // matrix of computed left vectors U
   matrixcf&       VT                     // transposed matrix of right vectors VT
   );
```

Parameters

jobz

[in] ENUM_SVD_Z enumeration value which determines the method for computing left and singular eigenvectors.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

© 2000-2025, MetaQuotes Ltd.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobz parameter.

When jobv is set to SVDZ_N, the left and right vectors are not computed. Only singular values are computed.

When jobv is set to SVDZ_A, the full matrices of the U and VT vectors are computed.

When the value is SVDZ_S, truncated matrices of vectors U and VT are computed.

ENUM_SVD_Z

An enumeration defining the way to compute left and right singular vectors.

ID	Description
SVDZ_N	Columns U or rows VT are not computed
SVDZ_A	All M columns of U or all N columns of VT are returned in arrays U and VT
SVDZ_S	The first min(M,N) columns of U or the first min(M,N) columns of VT are returned in arrays U and VT

See also

SingularValueDecompositionQR, SingularValueDecompositionQRPivot

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionQR

Singular Value Decomposition, QR algorithm. Considered a classical SVD algorithm (lapack function GESVD).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z       jobu,      // how to compute left vectors
   ENUM_SVD_Z       jobv,      // how to compute right vectors
   vector&          S,         // vector of computed singular values
   matrix&          U,         // matrix of computed left vectors U
   matrix&          VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z       jobu,      // how to compute left vectors
   ENUM_SVD_Z       jobv,      // how to compute right vectors
   vectorf&         S,         // vector of computed singular values
   matrixf&         U,         // matrix of computed left vectors U
   matrixf&         VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z       jobu,      // how to compute left vectors
   ENUM_SVD_Z       jobv,      // how to compute right vectors
   vector           S,         // vector of computed singular values
   matrixc          U,         // matrix of computed left vectors U
   matrixc          VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z       jobu,                   // how to compute left vectors
   ENUM_SVD_Z       jobv,                   // how to compute right vectors
   vectorf&         singular_values,        // vector of computed singular values
   matrixcf&        u,                      // matrix of computed left vectors U
   matrixcf&        vt                      // transposed matrix of right vectors VT
   );
```

Parameters

jobu

[in] ENUM_SVD_Z enumeration value defining how the left singular vectors should be computed.

jobv

© 2000-2025, MetaQuotes Ltd.

[in] ENUM_SVD_Z enumeration value defining how the right singular vectors should be computed.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobu and jobv parameters.

When set to SVDZ_N, the left (jobu) and right (jobv) vectors are not computed. Singular values are always computed.

When set to SVDZ_A, the full matrices of vectors U (jobu) or VT (jobv) are computed.

With the value SVDZ_S, truncated matrices of vectors U (jobu) or VT (jobv) are computed.

ENUM_SVD_Z

An enumeration defining the way to compute left and right singular vectors.

ID	Description
SVDZ_N	Columns U or rows VT are not computed
SVDZ_A	All M columns of U or all N columns of VT are returned in arrays U and VT
SVDZ_S	The first min(M,N) columns of U or the first min(M,N) columns of VT are returned in arrays U and VT

See also

SingularValueDecompositionDC, SingularValueDecompositionQRPivot

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionQRPivot

Singular Value Decomposition, QR with pivoting algorithm (lapack function GESVDQ).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // computation accuracy level
   ENUM_SVDQRP_P   jobp,     // use row reversal to compute
   ENUM_SVDQRP_R   jobr,     // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,     // how to compute left vectors
   ENUM_SVDQRP_V   jobv,     // how to compute right vectors
   vector&         S,        // vector of computed singular values
   matrix&         U,        // matrix of computed left vectors U
   matrix&         VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // computation accuracy level
   ENUM_SVDQRP_P   jobp,     // use row reversal to compute
   ENUM_SVDQRP_R   jobr,     // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,     // how to compute left vectors
   ENUM_SVDQRP_V   jobv,     // how to compute right vectors
   vectorf&        S,        // vector of computed singular values
   matrixf&        U,        // matrix of computed left vectors U
   matrixf&        VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // computation accuracy level
   ENUM_SVDQRP_P   jobp,     // use row reversal to compute
   ENUM_SVDQRP_R   jobr,     // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,     // how to compute left vectors
   ENUM_SVDQRP_V   jobv,     // how to compute right vectors
   vector&         S,        // vector of computed singular values
   matrixc&        U,        // matrix of computed left vectors U
   matrixc&        VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,                   // computation accuracy level
   ENUM_SVDQRP_P   jobp,                   // use row reversal to compute
   ENUM_SVDQRP_R   jobr,                   // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,                   // how to compute left vectors
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_SVDQRP_V    jobv,                  // how to compute right vectors
   vectorf&         singular_values,       // vector of computed singular values
   matrixcf&        u,                     // matrix of computed left vectors U
   matrixcf&        vt                     // transposed matrix of right vectors VT
   );
```

Parameters

joba

[in] ENUM_SVDQRP_A enumeration value defining the accuracy level of the SVD computation.

jobp

[in] ENUM_SVDQRP_P enumeration value defining the use of row reversal during the computation process.

jobr

[in] ENUM_SVDQRP_R enumeration value defining whether to transpose the triangular matrix R obtained as a result of the initial QR factorization.

jobu

[in] ENUM_SVDQRP_U enumeration value that determines how the left singular vectors should be computed.

jobv

[in] ENUM_SVDQRP_V enumeration value defining how the right singular vectors should be computed.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The number of matrix rows must not be less than the number of columns.

If both left and right singular vectors are computed, jobv must be set to SVDQRPV_R when jobu=SVDQRPU_R.

ENUM_SVDQRP_A

An enumeration that specifies the level of accuracy of the SVD computation.

© 2000-2025, MetaQuotes Ltd.

ID	Description
SVDQRP_A	The requested accuracy corresponds to the inverse error limited by epsilon. This is an aggressive level of truncation.
SVDQRPA_M	Similar to SVDQRP_A, but the truncation is softer. This is the average level of truncation.
SVDQRPA_H	High accuracy is required.

ENUM_SVDQRP_P

An enumeration that specifies whether to use row reversal during calculation.

ID	Description
SVDQRPP_P	The rows of A are ordered in descending order. Recommended for numerical reliability.
SVDQRPP_N	No row reversal.

ENUM_SVDQRP_R

An enumeration that specifies whether to transpose the triangular matrix R obtained as a result of the initial QR factorization.

ID	Description
SVDQRPR_T	After the initial rotated QR factorization, GESVD is applied to the transpose R**T of the calculated triangular factor R.
SVDQRPR_N	The triangular factor R is given as input to GESVD.

ENUM_SVDQRP_U

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDQRPU_A	All M left singular vectors are computed.
SVDQRPU_S	The min(M,N) left singular vectors are computed.
SVDQRPU_R	The numerical rank NUMRANK is determined and only the NUMRANK of the left singular vectors is computed.
SVDQRPU_F	N left singular vectors are returned.
SVDQRPU_N	Left singular vectors are not computed.

ENUM_SVDQRP_V

An enumeration defining how right singular vectors should be computed.

© 2000-2025, MetaQuotes Ltd.

ID	Description
SVDQRPV_A	All N right singular vectors are computed.
SVDQRPV_R	The numerical rank NUMRANK is defined and only the NUMRANK of right singular vectors are computed.
SVDQRPV_N	Right singular vectors are not computed.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionBisect

Singular Value Decomposition, bisection algorithm (lapack function GESVDX).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS    jobv,     // computation accuracy level
   ENUM_BLAS_RANGE     range,    // subset of computable singular values
   double              lower,    // lower limit of the subset
   double              upper,    // upper limit of the subset
   vector&             S,        // vector of computed singular values
   matrix&             U,        // matrix of computed left vectors U
   matrix&             VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS    jobv,     // computation accuracy level
   ENUM_BLAS_RANGE     range,    // subset of computable singular values
   double              lower,    // lower limit of the subset
   double              upper,    // upper limit of the subset
   vectorf&            S,        // vector of computed singular values
   matrixf&            U,        // matrix of computed left vectors U
   matrixf&            VT        // ransposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS    jobv,     // computation accuracy level
   ENUM_BLAS_RANGE     range,    // subset of computable singular values
   double              lower,    // lower limit of the subset
   double              upper,    // upper limit of the subset
   vector              S,        // vector of computed singular values
   matrixc             U,        // matrix of computed left vectors U
   matrixc             VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS    jobv,              // computation accuracy level
   ENUM_BLAS_RANGE     range,             // subset of computable singular values
   double              lower,             // lower limit of the subset
   double              upper,             // upper limit of the subset
   vectorf&            singular_values,   // vector of computed singular values
   matrixcf&           u,                 // matrix of computed left vectors U
   matrixcf&           vt                 // transposed matrix of right vectors VT
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Parameters

jobv

[in] ENUM_SVD_VECTORS enumeration value which determines the method for computing left and singular eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable singular values and vectors.

lower

[in] The lower limit of singular values subset; specified depending on the value of the range parameter.

upper

[in] The upper limit of singular values subset; specified depending on the value of the range parameter.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobuv* and *range* parameters.

When BLASRANGE_A is set, all singular values are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those singular values (and their vectors) that fall within the range of real values specified by the 'lower' and 'upper' parameters are computed.

With the BLASRANGE_I value, only those singular values (and their vectors) that fall within the range of integer indices specified by the 'lower' and 'upper' parameters are computed. For example, with lower=0 and upper=2, only the first three singular values are computed.

ENUM_SVD_VECTORS

An enumeration defining the way to compute left and right singular vectors.

© 2000-2025, MetaQuotes Ltd.

ID	Description
SVDVECTORS_N	Only singular values are computed, without vectors.
SVDVECTORS_U	Left singular vectors are computed.
SVDVECTORS_V	Right singular vectors are computed.
SVDVECTORS_UV	Left and right singular vectors are computed.

ENUM_BLAS_RANGE

An enumeration defining a subset of computable singular values and vectors.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	Singular or eigenvalues from IL to IU will be found

See also

SingularValueDecompositionDC, SingularValueDecompositionQR, SingularValueDecompositionQRPivot

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionJacobiHigh

Singular Value Decomposition, Jacobi high level algorithm (lapack function GEJSV).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // computation accuracy level
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   ENUM_SVDJH_R   jobr,            // define range of computable singular values
   ENUM_SVDJH_T   jobt,            // define whether to transpose when computing a squ
   ENUM_SVDJH_P   jobp,            // the possibility of structured perturbations to r
   vector&        S,               // vector of computed singular values
   matrix&        U,               // matrix of computed left vectors U
   matrix&        V,               // matrix of computed left vectors V
   vector&        work_results     // additional computation results
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // computation accuracy level
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   ENUM_SVDJH_R   jobr,            // define range of computable singular values
   ENUM_SVDJH_T   jobt,            // define whether to transpose when computing a squ
   ENUM_SVDJH_P   jobp,            // the possibility of structured perturbations to r
   vectorf&       S,               // vector of computed singular values
   matrixf&       U,               // U matrix of computed left vectors
   matrixf&       V,               // V matrix of computed left vectors
   vector&        work_results     // additional computation results
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // computation accuracy level
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   ENUM_SVDJH_R   jobr,            // define range of computable singular values
   ENUM_SVDJH_T   jobt,            // define whether to transpose when computing a squ
   ENUM_SVDJH_P   jobp,            // the possibility of structured perturbations to r
   vectorc&       S,               // vector of computed singular values
   matrixc&       U,               // U matrix of computed left vectors
   matrixc&       V,               // V matrix of computed left vectors
   vector&        work_results     // additional computation results
   );
```

© 2000-2025, MetaQuotes Ltd.

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A    joba,                  // computation accuracy level
   ENUM_SVDJH_U    jobu,                  // how to compute left vectors
   ENUM_SVDJH_V    jobv,                  // how to compute right vectors
   ENUM_SVDJH_R    jobr,                  // define range of computable singular valu
   ENUM_SVDJH_T    jobt,                  // define whether to transpose when computi
   ENUM_SVDJH_P    jobp,                  // the possibility of structured perturbati
   vectorf&        singular_values,       // vector of computed singular values S
   matrixcf&       u,                     // matrix of computed left vectors U
   matrixcf&       v,                     // matrix of computed left vectors V
   vectorf&        work_results           // additional computation results
   );
```

Parameters

joba

[in] ENUM_SVDJH_A enumeration value determining the accuracy level of the SVD computation.

jobu

[in] ENUM_SVDJH_U enumeration value that determines how the left singular vectors should be computed.

jobv

[in] ENUM_SVDJH_V enumeration value that determines how the right singular vectors should be computed.

jobr

[in] ENUM_SVDJH_R enumeration value defining the range of computable values

jobt

[in] ENUM_SVDJH_T enumeration value defining whether to transpose the matrix of it is square. If the matrix is non-square, this parameter is ignored.

jobp

[in] ENUM_SVDJH_P enumeration value defining the possibility of structured perturbations to remove denormalized values.

S

[out] Vector of singular values.

For work(1)/work(2) = one: the singular values of A. During the computation S contains Euclidean column norms of the iterated matrices in the array a.

For work(1)≠work(2): the singular values of A are (work(1)/work(2)) * S(1:n). This factored form is used if sigma_max(A) overflows or if small singular values have been saved from underflow by scaling the input matrix A.

jobr = 'R', some of the singular values may be returned as exact zeros obtained by 'setting to zero' because they are below the numerical rank threshold or are denormalized numbers.

U

© 2000-2025, MetaQuotes Ltd.

[out] Matrix of left singular vectors.

V

[out] Matrix of right singular vectors.

work_results

[out] Vector consisting of 7 statistics obtained as a result of the computation.

work(1) = scale = work(2)/work(1) is the scaling factor such that scale*sva(1:n) are the computed singular values of A. See the description of S.

work(2) = see the description of work(1).

work(3) = sconda is an estimate for the condition number of column equilibrated A. If joba = 'E' or 'G', sconda is an estimate of sqrt(||(R**t * R)**(-1)||_1). It is computed using ?pocon. It holds n**(-1/4) * sconda ≤ ||R**(-1)||_2 ≤ n**(1/4) * sconda, where R is the triangular factor from the QRF of A. However, if R is truncated and the numerical rank is determined to be strictly smaller than n, sconda is returned as -1, indicating that the smallest singular values might be lost.

If full SVD is needed, the following two condition numbers are useful for the analysis of the algorithm. They are provied for a user who is familiar with the details of the method.

work(4) = an estimate of the scaled condition number of the triangular factor in the first QR factorization.

work(5) = an estimate of the scaled condition number of the triangular factor in the second QR factorization.

The following two parameters are computed if jobt = 'T'. They are provided for a user who is familiar with the details of the method.

work(6) = the entropy of A**t*A :: this is the Shannon entropy of diag(A**t*A) / Trace(A**t*A) taken as point in the probability simplex.

work(7) = the entropy of A*A**t.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The number of matrix rows must not be less than the number of columns.

ENUM_SVDJH_A

An enumeration that specifies the level of accuracy of the SVD computation.

ID	Description
SVDJHA_C	Computation as with 'C' with an additional estimate of the condition number. It provides a realistic error bound.
SVDJHA_E	Computation as with SVDJHA_C' with an additional estimate of the condition number. It provides a realistic error bound.

© 2000-2025, MetaQuotes Ltd.

ID	Description
SVDJHA_F	Higher accuracy than the SVDJHA_C option.
SVDJHA_G	Computation as with SVDJHA_F with an additional estimate of the condition number.
SVDJHA_A	Small singular values are the noise and the matrix is treated as numerically rank deficient.
SVDJHA_R	Similar as in SVDJHA_A, but more accuracy.

ENUM_SVDJH_U

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJHU_U	N columns of U are returned in the array U.
SVDJHU_F	Full set of M left singular vectors is returned in the array U.
SVDJHU_N	U is not computed.

ENUM_SVDJH_V

An enumeration defining how right singular vectors should be computed.

ID	Description
SVDJHV_V	N columns of V are returned in the array V
SVDJHV_J	N columns of V are returned in the array V, but they are computed as the product of Jacobi rotations
SVDJHV_N	V is not computed

ENUM_SVDJH_R

An enumeration that defines the range of values to be computed.

ID	Description
SVDJHR_N	Do not kill small columns of c*A.
SVDJHR_R	RESTRICTED range for sigma(c*A). This option is recommended.

ENUM_SVDJH_T

An enumeration that specifies whether a matrix should be transposed if it is square.

© 2000-2025, MetaQuotes Ltd.

ID	Description
SVDJHT_T	Transpose if entropy test indicates possibly faster convergence of Jacobi process.
SVDJHT_N	Do not use transposition. Do not speculate.

ENUM_SVDJH_P

An enumeration that specifies the possibility of structured perturbations to remove denormalized values.

ID	Description
SVDJHP_P	Introduce perturbation.
SVDJHP_N	Do not perturb.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionJacobiLow

Singular Value Decomposition, Jacobi low level algorithm (lapack function GESVJ). The method computes small singular values and their singular vectors with much greater accuracy than other SVD routines in certain cases.

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   double         ctol,            // threshold for convergence if jobu='C'
   ulong          mv,              // number of first rows of matrix V if jobv='A'
   vector&        S,               // vector of computed singular values
   matrix&        U,               // matrix of computed left vectors U
   matrix&        V,               // matrix of computed left vectors V
   vector&        work_results     // additional computation results
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   double         ctol,            // threshold for convergence if jobu='C'
   ulong          mv,              // number of first rows of matrix V if jobv='A'
   vectorf&       S,               // vector of computed singular values
   matrixf&       U,               // U matrix of computed left vectors
   matrixf&       V,               // V matrix of computed left vectors
   vector&        work_results     // additional computation results
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   double         ctol,            // threshold for convergence if jobu='C'
   ulong          mv,              // number of first rows of matrix V if jobv='A'
   vector&        S,               // vector of computed singular values
   matrixc&       U,               // U matrix of computed left vectors
   matrixc&       V,               // V matrix of computed left vectors
   vector&        work_results     // additional computation results
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U    jobu,                  // how to compute left vectors
   ENUM_SVDJH_V    jobv,                  // how to compute right vectors
```

© 2000-2025, MetaQuotes Ltd.

```
   double          ctol,                  // threshold for convergence if jobu='C'
   ulong           mv,                    // number of first rows of matrix V if jobv
   vectorf&        singular_values,       // vector of computed singular values S
   matrixcf&       u,                     // matrix of computed left vectors U
   matrixcf&       v,                     // matrix of computed left vectors V
   vectorf&        work_results           // additional computation results
   );
```

Parameters

jobu

[in] ENUM_SVDJL_U enumeration value that determines how the left singular vectors should be computed.

jobv

[in] ENUM_SVDJL_V enumeration value defining how the right singular vectors should be computed.

ctol

[in] Convergence threshold if jobu=SVDJLU_C. For other values of 'jobu' the parameter is ignored.

mv

[in] Number of rows of matrix V to be computed if jobv=SVDJLV_A. For other values of 'jobv' the parameter is ignored.

S

[out] Vector of singular values.

Depending on the value scale = work(1), where scale is the scaling factor:

if scale = 1, S(1:n) contains the computed singular values of a. During the computation, sva contains the Euclidean column norms of the iterated matrices in the array a.

if scale ≠ 1, the singular values of a are scale*S(1:n), and this factored representation is due to the fact that some of the singular values of a might underflow or overflow.

U

[out] Matrix of left singular vectors.

V

[out] Matrix of right singular vectors (non-transposed).

work_results

[out] Vector consisting of 7 statistics obtained as a result of the computation.

work(1) = scale is the scaling factor such that scale*S(1:n) are the computed singular values of A. See the description of S).

work(2) is the number of the computed nonzero singular value.

work(3) is the number of the computed singular values that are larger than the underflow threshold.

work(4) is the number of sweeps of Jacobi rotations needed for numerical convergence.

© 2000-2025, MetaQuotes Ltd.

work(5) = max_{i.NE.j} |COS(A(:,i),A(:,j))| in the last sweep. This is useful information in cases when ?gesvj did not converge, as it can be used to estimate whether the output is still useful and for post festum analysis.

work(6) is the largest absolute value over all sines of the Jacobi rotation angles in the last sweep. It can be useful in a post festum analysis.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The number of matrix rows must not be less than the number of columns.

ENUM_SVDJL_U

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJLU_U	The left singular vectors corresponding to the nonzero singular values are computed and returned in the leading columns of A
SVDJLU_C	Analogous to SVDJLU_U, except that user can control the level of numerical orthogonality of the computed left singular vectors.
SVDJLU_N	The matrix U is not computed.

ENUM_SVDJL_V

An enumeration defining how right singular vectors should be computed.

ID	Description
SVDJLV_V	The matrix V is computed
SVDJLV_A	The Jacobi rotations are applied to the MV-by-N matrix V.
SVDJLV_N	The matrix V is not computed.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagDC

Singular Value Decomposition, divide-and-conquer algorithm for bidiagonal matrices (lapack function BDSDC).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagDC(
   ENUM_SVDBIDIAG_Z    jobz,       // how to compute left vectors
   ENUM_BLAS_RANGE     range,      // subset of computed singular values
   double              lower,      // lower limit of the subset
   double              upper,      // upper limit of the subset
   vector&             S,          // vector of computed singular values
   matrix&             U,          // U matrix of computed left vectors
   matrix&             VT          // VT transposed matrix of right vectors
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagDC(
   ENUM_SVDBIDIAG_Z    jobz,       // how to compute left vectors
   ENUM_BLAS_RANGE     range,      // subset of computed singular values
   double              lower,      // lower limit of the subset
   double              upper,      // upper limit of the subset
   vectorf&            S,          // vector of computed singular values
   matrixf&            U,          // U matrix of computed left vectors
   matrixf&            VT          // VT transposed matrix of right vectors
   );
```

Parameters

jobz

[in] ENUM_SVDBIDIAG_Z enumeration value that determines how the left singular vectors should be computed.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable singular values and vectors.

lower

[in] The lower limit of singular values subset; specified depending on the value of the range parameter.

upper

[in] The upper limit of singular values subset; specified depending on the value of the range parameter.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

© 2000-2025, MetaQuotes Ltd.

V

[out] Transposed matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the jobz and range parameters.

When BLASRANGE_A is set, all singular values are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those singular values (and their vectors) that fall within the range of real values specified by the 'lower' and 'upper' parameters are computed.

With the BLASRANGE_I value, only those singular values (and their vectors) that fall within the range of integer indices specified by the 'lower' and 'upper' parameters are computed. For example, with lower=0 and upper=2, only the first three singular values are computed.

A bidiagonal matrix is a square matrix with non-zero main diagonal and one of the sub-diagonals.

Upper bidiagonal matrix

```
[[x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x],
 [0, 0, 0, 0, x]]
```

Lower bidiagonal matrix

```
[[x, 0, 0, 0, 0],
 [x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x]]
```

ENUM_SVDBIDIAG_Z

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJOBZ_V	Compute singular values and singular vectors.
SVDJOBZ_N	Compute singular values only.

ENUM_BLAS_RANGE

© 2000-2025, MetaQuotes Ltd.

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagBisect

Singular Value Decomposition, bisection algorithm for bidiagonal matrices (lapack function BDSVDX).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagBisect(
   ENUM_SVDBIDIAG_Z    jobz,       // how to compute left vectors
   ENUM_BLAS_RANGE     range,      // subset of computed singular values
   double              lower,      // lower limit of the subset
   double              upper,      // upper limit of the subset
   vector&             S,          // vector of computed singular values
   matrix&             U,          // U matrix of computed left vectors
   matrix&             VT          // VT transposed matrix of right vectors
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagBisect(
   ENUM_SVDBIDIAG_Z    jobz,       // how to compute left vectors
   ENUM_BLAS_RANGE     range,      // subset of computed singular values
   double              lower,      // lower limit of the subset
   double              upper,      // upper limit of the subset
   vectorf&            S,          // vector of computed singular values
   matrixf&            U,          // U matrix of computed left vectors
   matrixf&            VT          // VT transposed matrix of right vectors
   );
```

Parameters

jobz

[in] ENUM_SVDBIDIAG_Z enumeration value that determines how the left singular vectors should be computed.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable singular values and vectors.

lower

[in] The lower limit of singular values subset; specified depending on the value of the range parameter.

upper

[in] The upper limit of singular values subset; specified depending on the value of the range parameter.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

© 2000-2025, MetaQuotes Ltd.

V

[out] Transposed matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

When BLASRANGE_A is set, all singular values are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those singular values (and their vectors) that fall within the range of real values specified by the 'lower' and 'upper' parameters are computed.

With the BLASRANGE_I value, only those singular values (and their vectors) that fall within the range of integer indices specified by the 'lower' and 'upper' parameters are computed. For example, with lower=0 and upper=2, only the first three singular values are computed.

A bidiagonal matrix is a square matrix with non-zero main diagonal and one of the sub-diagonals.

Upper bidiagonal matrix

```
[[x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x],
 [0, 0, 0, 0, x]]
```

Lower bidiagonal matrix

```
[[x, 0, 0, 0, 0],
 [x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x]]
```

ENUM_SVDBIDIAG_Z

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJOBZ_V	Compute singular values and singular vectors.
SVDJOBZ_N	Compute singular values only.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

© 2000-2025, MetaQuotes Ltd.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagQR

Computes the singular value decomposition of a general matrix that has been reduced to bidiagonal form by the method ReduceToBidiagonal. Lapack function BDSQR.

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrix&          Q,          // orthogonal matrix Q
   matrix&          PT,         // transposed matrix P
   vector&          S,          // vector of computed singular values
   matrix&          U,          // matrix U of computed left vectors
   matrix&          VT          // transposed matrix V of right vectors
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrix&          Q,          // orthogonal matrix Q
   matrix&          PT,         // transposed matrix P
   vectorf&         S,          // vector of computed singular values
   matrixf&         U,          // matrix U of computed left vectors
   matrixf&         VT          // transposed matrix V of right vectors
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrix&          Q,          // orthogonal matrix Q
   matrix&          PH,         // Hermitian conjugated matrix P
   vector           S,          // vector of computed singular values
   matrixc          U,          // matrix U of computed left vectors
   matrixc          VH          // Hermitian conjugated matrix V of right vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrixf&         Q,          // orthogonal matrix Q
   matrixf&         PH,         // Hermitian conjugated matrix P
   vectorf&         S,          // vector of computed singular values
   matrixcf&        U,          // matrix U of computed left vectors
   matrixcf&        VH          // Hermitian conjugated matrix V of right vectors
   );
```

Parameters

Q

[in] Orthogonal matrix Q produced by method ReflectBidiagonalToQP. If matrix Q has zero size, then left vectors U are not calculated.

PT

© 2000-2025, MetaQuotes Ltd.

[in] Transposed (or hermitian conjugated) matrix P produced by method ReflectBidiagonalToQP. If matrix PT has zero size, then right vectors VT are not calculated.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

BDSQR computes the singular values and, optionally, the right and/or left singular vectors from the singular value decomposition (SVD) of a N-by-N (upper or lower) bidiagonal matrix B using the implicit zero-shift QR algorithm. The SVD of B has the form

B = Q * S * P**T

where S is the diagonal matrix of singular values, Q is an orthogonal matrix of left singular vectors, and P is an orthogonal matrix of right singular vectors. If left singular vectors are requested, this subroutine actually returns U*Q instead of Q, and, if right singular vectors are requested, this subroutine returns P**T*VT instead of P**T, for given real input matrices U and VT. When U and VT are the orthogonal matrices that reduce a general matrix A to bidiagonal form: A = U*B*VT, as computed by GEBRD, then

A = (U*Q) * S * (P**T*VT)

is the SVD of A.

© 2000-2025, MetaQuotes Ltd.

Eigen Values

The section features functions for computing eigenvalues and eigenvectors. It describes methods for solving standard linear algebra problems using the LAPACK library algorithms. These functions are efficient for matrix analysis, diagonalization, system stabilization, and other tasks.

- **EigenSolver**: The function is designed to compute the eigenvalues and eigenvectors of an arbitrary square matrix using the classical algorithm represented by the GEEV lapack function. This method is applied to a wide range of matrices, allowing the decomposition of matrices into their eigenvalues and eigenvectors.
- **EigenSymmetricDC**: The function for computing eigenvalues and eigenvectors of symmetric or Hermitian matrices using the divide-and-conquer algorithm. The lapack functions SYEVD and HEEVD enable the efficient handling of symmetric or Hermitian matrices, providing faster and more accurate processing of such matrices.

Function	Action
EigenSolver	Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (lapack function GEEV).
EigenSolverX	Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GEEVX).
EigenSolverShur	Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (lapack function GEES). See also Schur decomposition.
EigenSolver2	Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (lapack function GGEV).
EigenSolver2X	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GGEVX). Both matrices must be the same size.
EigenSolver2Shur	Compute a pair of ordinary square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES).
EigenSolver2Blocked	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices using a block algorithm (lapack function GGEV3). Both matrices must be the same size. The method parameters are exactly the same as EigenSolver2.

© 2000-2025, MetaQuotes Ltd.

Function	Action
EigenSolver2ShurBlocked	Compute a pair of regular square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES3).
EigenSymmetricDC	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the divide-and-conquer algorithm (lapack functions SYEVD, HEEVD).
EigenSymmetricQR	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the QR algorithm (lapack functions SYEV, HEEV).
EigenSymmetricRobust	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (lapack functions SYEVR, HEEVR).
EigenSymmetricBisect	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the bisection algorithm (lapack functions SYEVX, HEEVX).

© 2000-2025, MetaQuotes Ltd.

General Matrices

Functions for calculating eigenvalues and eigenvectors of a square matrix using classical algorithms. It provides various methods for working with both real and complex matrices, allowing you to solve linear algebra problems with a choice of methods for calculating eigenvectors.

Function	Action
EigenSolver	Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (lapack function GEEV).
EigenSolverX	Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GEEVX).
EigenSolverShur	Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (lapack function GEES). See also Schur decomposition.
EigenSolver2	Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (lapack function GGEV).
EigenSolver2X	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GGEVX). Both matrices must be the same size.
EigenSolver2Shur	Compute a pair of ordinary square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES).
EigenSolver2Blocked	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices using a block algorithm (lapack function GGEV3). Both matrices must be the same size. The method parameters are exactly the same as EigenSolver2.
EigenSolver2ShurBlocked	Compute a pair of regular square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES3).

© 2000-2025, MetaQuotes Ltd.

EigenSolver

Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (lapack function GEEV).

Computing for type matrix<double>

```
bool  matrix::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right eigenvecto
   vectorc&              eigen_values,         // vector of computed eigenvalues
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right eigenvecto
   vectorcf&             eigen_values,         // vector of computed eigenvalues
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right vectors
   vectorc&              eigen_values,         // vector of computed eigenvalues
   matrixc&              left_eigenvectors,    // matrix of computed left vectors
   matrixc&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right vectors
   vectorcf&             eigen_values,         // vector of computed eigenvalues
   matrixcf&             left_eigenvectors,    // matrix of computed left vectors
   matrixcf&             right_eigenvectors    // matrix of computed right vectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

EV

[out] Vector of eigenvalues.

left_eigenvectors

[out] Matrix of left eigenvectors.

© 2000-2025, MetaQuotes Ltd.

right_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobv parameter.

If EIGVECTORS_N is set, the left and right vectors are not computed. Only eigenvalues are computed.

With EIGVECTORS_L, only left eigenvectors are computed, right eigenvectors are not computed.

When EIGVECTORS_R is set, only the right eigenvectors are computed, the left vectors are not computed.

With EIGVECTORS_LR, the left and right eigenvectors are computed, Eigenvalues are always computed.

Real (non-complex) matrices can have a complex solution. Therefore, the vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

ENUM_EIG_VECTORS

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are calculated, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000-2025, MetaQuotes Ltd.

EigenSolverX

Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GEEVX).

Computing for type matrix<double>

```
bool  matrix::EigenSolverX(
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right a
   ENUM_EIG_SENSE        sense,                // determines computation of recipro
   vectorc&              eigen_values,         // vector of computed eigenvalues
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right vectors
   matrix&               shur_matrix,          // balanced matrix in Schur form
   long&                 ilo,                  // subscript of balanced matrix
   long&                 ihi,                  // superscript of balanced matrix
   vector&               scale,                // details of permutations and scali
   double&               ab_norm,              // 1-norm of balanced matrix
   vector&               rconde,               // vector of reciprocal condition nu
   vector&               rcondv                // vector of reciprocal condition nu
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolverX(
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right a
   ENUM_EIG_SENSE        sense,                // determines computation of recipro
   vectorcf&             eigen_values,         // vector of computed eigenvalues
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right vectors
   matrixf&              shur_matrix,          // balanced matrix in Schur form
   long&                 ilo,                  // subscript of balanced matrix
   long&                 ihi,                  // superscript of balanced matrix
   vectorf&              scale,                // details of permutations and scali
   float&                ab_norm,              // 1-norm of balanced matrix
   vectorf&              rconde,               // vector of reciprocal condition nu
   vectorf&              rcondv                // vector of reciprocal condition nu
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolverX(
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right a
   ENUM_EIG_SENSE        sense,                // determines computation of recipro
   vectorc&              eigen_values,         // vector of computed eigenvalues
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixc&              left_eigenvectors,       // matrix of computed left vectors
   matrixc&              right_eigenvectors       // matrix of computed right vectors
   matrixc&              shur_matrix,             // balanced matrix in Schur form
   long&                 ilo,                     // subscript of balanced matrix
   long&                 ihi,                     // superscript of balanced matrix
   vector&               scale,                   // details of permutations and scali
   double&               ab_norm,                 // 1-norm of balanced matrix
   vector&               rconde,                  // vector of reciprocal condition nu
   vector&               rcondv                   // vector of reciprocal condition nu
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolverX(
   ENUM_EIG_BALANCE      balance,                 // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                    // determines computation of right a
   ENUM_EIG_SENSE        sense,                   // determines computation of recipro
   vectorcf&             eigen_values,            // vector of computed eigenvalues
   matrixcf&             left_eigenvectors,       // matrix of computed left vectors
   matrixcf&             right_eigenvectors       // matrix of computed right vectors
   matrixcf&             shur_matrix,             // balanced matrix in Schur form
   long&                 ilo,                     // subscript of balanced matrix
   long&                 ihi,                     // superscript of balanced matrix
   vectorf&              scale,                   // details of permutations and scali
   float&                ab_norm,                 // 1-norm of balanced matrix
   vectorf&              rconde,                  // vector of reciprocal condition nu
   vectorf&              rcondv                   // vector of reciprocal condition nu
   );
```

Parameters

balance

[in] Value from the ENUM_EIG_BALANCE enumeration which determines the need and method for balancing the input matrix; it is used to improve the conditioning of the eigenvalues and eigenvectors.

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

sense

[in] Value from the ENUM_EIG_SENSE enumeration determining the need to compute reciprocal condition numbers.

eigen_values

[out] Vector of eigenvalues.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

© 2000-2025, MetaQuotes Ltd.

[out] Matrix of right eigenvectors.

shur_matrix

[out] Balanced matrix in Schur form; the matrix is not filled if neither left nor right eigenvectors are computed.

ilo

[out] Subscript of the balanced matrix; the matrix is not filled if no balancing is applied.

ihi

[out] Superscript of the balanced matrix; the matrix is not filled if no balancing is applied.

scale

[out] Vector of details of permutations and scaling when balancing the input matrix.

Details of the permutations and scaling factors applied when balancing A.

If P(j) is the index of the row and column interchanged with row and column j, and D(j) is the scaling factor applied to row and column j, then

scale(j) = P(j), for j = 1,...,ilo-1

= D(j), for j = ilo,...,ihi

= P(j) for j = ihi+1,..., n.

The order in which the interchanges are made is n to ihi+1, then 1 to ilo-1.

ab_norm

[out] 1-norm of the balanced matrix (the maximum of the sum of absolute values of elements in any of the matrix columns).

rconde

[out] Vector of reciprocal condition numbers for each eigenvalue; it is computed if the 'sense' parameter is set to 'E' or 'B'.

rcondv

[out] Vector of reciprocal condition numbers for each eigenvector; it is computed if the 'sense' parameter is set to 'V' or 'B.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Input matrix balancing depends on the value of the '*balance*' parameter.

ENUM_EIG_BALANCE

An enumeration defining the need to compute eigenvectors.

© 2000-2025, MetaQuotes Ltd.

ID	Description
EIGBALANCE_N	Do not diagonally scale or permute
EIGBALANCE_P	Perform permutations to make the matrix more nearly upper triangular. Do not diagonally scale
EIGBALANCE_S	Diagonally scale the matrix. Do not permute
EIGBALANCE_B	Both diagonally scale and permute

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

ENUM_EIG_SENSE

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGSENSE_N	None of reciprocal condition numbers are computed
EIGSENSE_E	Computed for eigenvalues only
EIGSENSE_V	Computed for right eigenvectors only
EIGSENSE_B	Computed for eigenvalues and right eigenvectors

© 2000-2025, MetaQuotes Ltd.

EigenSolverShur

Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (lapack function GEES). See also Schur decomposition.

Computing for type matrix<double>

```
bool  matrix::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // method to compute Shur vectors
   vectorc&              eigen_values,           // vector of computed eigenvalues
   matrix&               shur_matrix,            // matrix in Schur form
   matrix&               shur_vectors            // matrix of Schur vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // method to compute Shur vectors
   vectorcf&             eigen_values,           // vector of computed eigenvalues
   matrixf&              shur_matrix,            // matrix in Schur form
   matrixf&              shur_vectors            // matrix of Schur vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // method to compute Shur vectors
   vectorc&              eigen_values,           // vector of computed eigenvalues
   matrixc&              shur_matrix,            // matrix in Schur form
   matrixc&              shur_vectors            // matrix of Schur vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // method to compute Shur vectors
   vectorcf&             eigen_values,           // vector of computed eigenvalues
   matrixcf&             shur_matrix,            // matrix in Schur form
   matrixcf&             shur_vectors            // matrix of Schur vectors
   );
```

Parameters

jobvs

[in] Value from the ENUM_EIG_SHUR enumeration, which defines the method for computing Shur vectors.

eigen_values

[out] Vector of eigenvalues.

shur_matrix

[out] Upper triangular Schur matrix (Schur form for the input matrix).

© 2000-2025, MetaQuotes Ltd.

shur_vectors

[out] Matrix of Schur vectors; it is not computed if the *jobvs* parameter is set to N.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the *jobvs* parameter values.

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

EigenSolverShur

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGSHUR_N	Schur vectors are not computed
EIGSHUR_V	Schur vectors are computed

© 2000-2025, MetaQuotes Ltd.

EigenSolver2

Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (lapack function GGEV). Both matrices must be the same size.

Computing for type matrix<double>

```
bool  matrix::EigenSolver2(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorc&              alpha,                // vector of computed eigenvalues
   vector&               beta,                 // vector of eigenvalue divisors
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver2(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorf&              beta,                 // vector of eigenvalue divisors
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolver2(
   matrixc&              B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorc&              alpha,                // vector of computed eigenvalues
   vectorc&              beta,                 // vector of eigenvalue divisors
   matrixc&              left_eigenvectors,    // matrix of computed left vectors
   matrixc&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver2(
   matrixcf&             B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorcf&             beta,                 // vector of eigenvalue divisors
   matrixcf&             left_eigenvectors,    // matrix of computed left vectors
   matrixcf&             right_eigenvectors    // matrix of computed right vectors
   );
```

Parameters

© 2000-2025, MetaQuotes Ltd.

B

[in] The second matrix in the pair.

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigen value divisors.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobv parameter.

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

A generalized eigenvalue for a pair of matrices (A,B) is a scalar lambda or a ratio alpha/beta = lambda, such that A - lambda*B is singular. It is usually represented as the pair (alpha,beta), as there is a reasonable interpretation for beta=0, and even for both being zero.

The right eigenvector v(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies

A * v(j) = lambda(j) * B * v(j).

The left eigenvector u(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies

u(j)**H * A = lambda(j) * u(j)**H * B .

© 2000-2025, MetaQuotes Ltd.

where u(j)**H is the conjugate-transpose of u(j).

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

© 2000-2025, MetaQuotes Ltd.

EigenSolver2X

Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GGEVX). Both matrices must be the same size.

Optionally, it also computes a balancing transformation to improve the conditioning of the eigenvalues and eigenvectors (ILO, IHI, LSCALE, RSCALE, ABNRM, and BBNRM), reciprocal condition numbers for the eigenvalues (RCONDE), and reciprocal condition numbers for the right eigenvectors (RCONDV).

Computing for type matrix<double>

```
bool  matrix::EigenSolver2X(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right a
   ENUM_EIG_SENSE        sense,                // determines computation of recipro
   vectorc&              alpha,                // vector of computed eigenvalues
   vector&               beta,                 // vector of eigenvalue divisors
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right vectors
   matrix&               shur_matrix1,         // the first part of the real Schur
   matrix&               shur_matrix2,         // the second part of the real Schu
   long&                 ilo,                  // subscript of balanced matrix
   long&                 ihi,                  // superscript of balanced matrix
   vector&               lscale,               // details of the permutations and s
   vector&               rscale,               // details of the permutations and s
   double&               ab_norm,              // one-norm of balanced input matrix
   double&               bb_norm,              // one-norm of balanced second matri
   vector&               rconde,               // vector of reciprocal condition nu
   vector&               rcondv                // vector of reciprocal condition nu
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver2X(
   matrixf&              B,                    // second matrix in the pair
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right a
   ENUM_EIG_SENSE        sense,                // determines computation of recipro
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorf&              beta,                 // vector of eigenvalue divisors
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right vectors
   matrixf&              shur_matrix1,         // the first part of the real Schur
   matrixf&              shur_matrix2,         // the second part of the real Schu
   long&                 ilo,                  // subscript of balanced matrix
   long&                 ihi,                  // superscript of balanced matrix
   vectorf&              lscale,               // details of the permutations and s
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf&              rscale,                 // details of the permutations and s
   float&                ab_norm,                // one-norm of balanced input matrix
   float&                bb_norm,                // one-norm of balanced second matri
   vectorf&              rconde,                 // vector of reciprocal condition nu
   vectorf&              rcondv                  // vector of reciprocal condition nu
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolver2X(
   matrixc&              B,                      // second matrix in the pair
   ENUM_EIG_BALANCE      balance,                // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                   // determines computation of right a
   ENUM_EIG_SENSE        sense,                  // determines computation of recipro
   vectorc&              alpha,                  // vector of computed eigenvalues
   vectorc&              beta,                   // vector of eigenvalue divisors
   matrixc&              left_eigenvectors,      // matrix of computed left vectors
   matrixc&              right_eigenvectors      // matrix of computed right vectors
   matrixc&              shur_matrix1,           // the first part of the real Schur
   matrixc&              shur_matrix2,           // the second part of the real Schur
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vector&               lscale,                 // details of the permutations and s
   vector&               rscale,                 // details of the permutations and s
   double&               ab_norm,                // one-norm of balanced input matrix
   double&               bb_norm,                // one-norm of balanced second matri
   vector&               rconde,                 // vector of reciprocal condition nu
   vector&               rcondv                  // vector of reciprocal condition nu
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver2X(
   matrixcf&             B,                      // second matrix in the pair
   ENUM_EIG_BALANCE      balance,                // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                   // determines computation of right a
   ENUM_EIG_SENSE        sense,                  // determines computation of recipro
   vectorcf&             alpha,                  // vector of computed eigenvalues
   vectorf&              beta,                   // vector of eigenvalue divisors
   matrixcf&             left_eigenvectors,      // matrix of computed left vectors
   matrixcf&             right_eigenvectors      // matrix of computed right vectors
   matrixcf&             shur_matrix1,           // the first part of the real Schur
   matrixcf&             shur_matrix2,           // the second part of the real Schur
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vectorf&              lscale,                 // details of the permutations and s
   vectorf&              rscale,                 // details of the permutations and s
   float&                ab_norm,                // one-norm of balanced input matrix
```

© 2000-2025, MetaQuotes Ltd.

```
   float&                bb_norm,                // one-norm of balanced second matri
   vectorf&              rconde,                 // vector of reciprocal condition nu
   vectorf&              rcondv                  // vector of reciprocal condition nu
   );
```

Parameters

B

[in] The second matrix in the pair.

balance

[in] Value from the ENUM_EIG_BALANCE enumeration which determines the need and method for balancing the input matrix; it is used to improve the conditioning of the eigenvalues and eigenvectors.

jobv

[in] Value from the ENUM_EIG_VECTORS enumeration which determines the method for computing left and right eigenvectors.

sense

[in] Value from the ENUM_EIG_SENSE enumeration which determines the need to compute reciprocal condition numbers.

eigen_values

[out] Vector of eigenvalues.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

[out] Matrix of right eigenvectors.

shur_matrix1, shur_matrix2

[out] 2 parts of balanced matrix in Schur form; the matrix is not filled if neither left nor right eigenvectors are computed.

ilo

[out] Subscript of the balanced matrix; the matrix is not filled if no balancing is applied.

ihi

[out] Superscript of the balanced matrix; the matrix is not filled if no balancing is applied.

lscale

[out] Vector contains details of the permutations and scaling factors applied to the left side of A and B.

If PL(j) is the index of the row interchanged with row j, and DL(j) is the scaling factor applied to row j, then

lscale(j) = PL(j), for j = 1,..., ilo-1

= DL(j), for j = ilo,...,ihi

= PL(j) for j = ihi+1,..., n.

© 2000-2025, MetaQuotes Ltd.

The order in which the interchanges are made is n to ihi+1, then 1 to ilo-1.

rscale

[out] Vector contains details of the permutations and scaling factors applied to the right side of A and B.

If PR(j) is the index of the column interchanged with column j, and DR(j) is the scaling factor applied to column j, then

rscale(j) = PR(j), for j = 1,..., ilo-1

= DR(j), for j = ilo,...,ihi

= PR(j) for j = ihi+1,..., n.

The order in which the interchanges are made is n to ihi+1, then 1 to ilo-1.

ab_norm

[out] One-norm of the balanced input matrix (the maximum of the sum of absolute values of elements in any of the matrix columns).

bb_norm

[out] One-norm of balanced second matrix B.

rconde

[out] Vector of reciprocal condition numbers for each eigenvalue; it is computed if the 'sense' parameter is set to 'E' or 'B'.

rcondv

[out] Vector of reciprocal condition numbers for each eigenvector; it is computed if the 'sense' parameter is set to 'V' or 'B'.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Input matrices balancing depends on the value of the *balance* parameter.

ENUM_EIG_BALANCE

An enumeration that specifies whether the matrices should be balanced.

ID	Description
EIGBALANCE_N	Do not diagonally scale or permute.
EIGBALANCE_P	Perform permutations to make the matrix more nearly upper triangular. Do not diagonally scale.
EIGBALANCE_S	Diagonally scale the matrix. Do not permute.
EIGBALANCE_B	Both diagonally scale and permute.

© 2000-2025, MetaQuotes Ltd.

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

ENUM_EIG_SENSE

An enumeration determining the need to compute reciprocal condition numbers.

ID	Description
EIGSENSE_N	None of reciprocal condition numbers are computed.
EIGSENSE_E	Computed for eigenvalues only.
EIGSENSE_V	Computed for right eigenvectors only.
EIGSENSE_B	Computed for eigenvalues and right eigenvectors.

© 2000-2025, MetaQuotes Ltd.

EigenSolver2Shur

Compute a pair of ordinary square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES).

Computes the generalized eigenvalues, the generalized real/complex Schur form (S,T), optionally, the left and/or right matrices of Schur vectors (vsl and vsr) for a pair of n-by-n real/complex nonsymmetric matrices (A,B). This gives the generalized Schur factorization:

(A,B) = (vsl*S *vsrH, vsl*T*vsrH)

Optionally, it also orders the eigenvalues so that a selected cluster of eigenvalues appears in the leading diagonal blocks of the upper quasi-triangular matrix S and the upper triangular matrix T. The leading columns of vsl and vsr then form an orthonormal/unitary basis for the corresponding left and right eigenspaces (deflating subspaces).

Computing for type matrix<double>

```
bool  matrix::EigenSolver2Shur(
   matrix&                 B,         // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,     // method to compute left and right vectors
   vector&                 alpha,     // vector of computed eigenvalues
   vector&                 beta,      // vector of eigenvalue divisors
   matrix&                 shur_s,    // matrix S in Schur form
   matrix&                 shur_t,    // matrix T in Schur form
   matrix&                 vsl,       // matrix of left Schur vectors vsl
   matrix&                 vsr        // matrix of right Schur vectors vsr
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver2Shur(
   matrixf&                B,         // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,     // method to compute left and right vectors
   vectorcf&               alpha,     // vector of computed eigenvalues
   vectorf&                beta,      // vector of eigenvalue divisors
   matrixf&                shur_s,    // matrix S in Schur form
   matrixf&                shur_t,    // matrix T in Schur form
   matrixf&                vsl,       // matrix of left Schur vectors vsl
   matrixf&                vsr        // matrix of right Schur vectors vsr
   );
```

Computing for type matrix<complex>

```
bool  matrix::EigenSolver2Shur(
   matrixc&                B,         // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,     // method to compute left and right vectors
   vectorc&                alpha,     // vector of computed eigenvalues
   vectorc&                beta,      // vector of eigenvalue divisors
   matrixc&                shur_s,    // matrix S in Schur form
   matrixc&                shur_t,    // matrix T in Schur form
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixc&                vsl,       // matrix of left Schur vectors vsl
   matrixc&                vsr        // matrix of right Schur vectors vsr
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSolver2Shur(
   matrixcf&               B,         // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,     // method to compute left and right vectors
   vectorcf&               alpha,     // vector of computed eigenvalues
   vectorcf&               beta,      // vector of eigenvalue divisors
   matrixcf&               shur_s,    // matrix S in Schur form
   matrixcf&               shur_t,    // matrix T in Schur form
   matrixcf&               vsl,       // matrix of left Schur vectors vsl
   matrixcf&               vsr        // matrix of right Schur vectors vsr
   );
```

Parameters

B

[out] The second matrix in the pair.

jobvs

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigenvalue divisors.

shur_s

[out] Matrix S, block upper triangular Schur matrix (Schur form for the input matrix).

shur_t

[out] Matrix T, block upper triangular Schur matrix (Schur form for the second matrix in the pair).

vsl

[out] Matrix of left Schur vectors.

vsr

[out] Matrix of right Schur vectors.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Computation depends on the *jobvs* parameter values.

© 2000-2025, MetaQuotes Ltd.

The second matrix in the pair must be the same size as the first (input) one.

Real (non-complex) matrices can have a complex solution. Therefore, the vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

EigenSolverShur

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000-2025, MetaQuotes Ltd.

EigenSolver2Blocked

Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices using a block algorithm (lapack function GGEV3). Both matrices must be the same size. The method parameters are exactly the same as EigenSolver2.

Computing for type matrix<double>

```
bool  matrix::EigenSolver2Blocked(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorc&              alpha,                // vector of computed eigenvalues
   vector&               beta,                 // vector of eigenvalue divisors
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver2Blocked(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorf&              beta,                 // vector of eigenvalue divisors
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complex>

```
bool  matrix::EigenSolver2Blocked(
   matrixc&              B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorc&              alpha,                // vector of computed eigenvalues
   vectorc&              beta,                 // vector of eigenvalue divisors
   matrixc&              left_eigenvectors,    // matrix of computed left vectors
   matrixc&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver2(
   matrixcf&             B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorcf&             beta,                 // vector of eigenvalue divisors
   matrixcf&             left_eigenvectors,    // matrix of computed left vectors
   matrixcf&             right_eigenvectors    // matrix of computed right vectors
   );
```

Parameters

© 2000-2025, MetaQuotes Ltd.

B

[out] The second matrix in the pair.

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigenvalue divisors.

left_eigenvectors

[out] Matrix of left eigenvectors.

righeft_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Computation depends on the value of the jobv parameter.

A generalized eigenvalue for a pair of matrices (A,B) is a scalar lambda or a ratio alpha/beta = lambda, such that A - lambda*B is singular. It is usually represented as the pair (alpha,beta), as there is a reasonable interpretation for beta=0, and even for both being zero.

The right eigenvector v(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies:

A * v(j) = lambda(j) * B * v(j).

The left eigenvector u(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies:

u(j)**H * A = lambda(j) * u(j)**H * B .

where u(j)**H is the conjugate-transpose of u(j).

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.

© 2000-2025, MetaQuotes Ltd.

ID	Description
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000-2025, MetaQuotes Ltd.

EigenSolver2ShurBlocked

Compute a pair of regular square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES3).

Computes the generalized eigenvalues, the generalized real/complex Schur form (S,T), optionally, the left and/or right matrices of Schur vectors (VSL and VSR) for a pair of n-by-n real/complex nonsymmetric matrices (A,B). This gives the generalized Schur factorization:

(A,B) = (vsl*S *vsrH, vsl*T*vsrH)

Optionally, it also orders the eigenvalues so that a selected cluster of eigenvalues appears in the leading diagonal blocks of the upper quasi-triangular matrix S and the upper triangular matrix T. The leading columns of vsl and vsr then form an orthonormal/unitary basis for the corresponding left and right eigenspaces (deflating subspaces).

Computing for type matrix<double>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrix&                B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vectorc&               alpha,     // vector of computed eigenvalues
   vector&                beta,      // vector of eigenvalue divisors
   matrix&                shur_s,    // matrix S in Schur form
   matrix&                shur_t,    // matrix T in Schur form
   matrix&                vsl,       // matrix of left Schur vectors VSL
   matrix&                vsr        // matrix of right Schur vectors VSR
   );
```

Computing for type matrix<float>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrixf&               B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vectorcf&              alpha,     // vector of computed eigenvalues
   vectorf&               beta,      // vector of eigenvalue divisors
   matrixf&               shur_s,    // matrix S in Schur form
   matrixf&               shur_t,    // matrix T in Schur form
   matrixf&               vsl,       // matrix of left Schur vectors VSL
   matrixf&               vsr        // matrix of right Schur vectors VSR
   );
```

Computing for type matrix<complex>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrixc&               B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vectorc&               alpha,     // vector of computed eigenvalues
   vectorc&               beta,      // vector of eigenvalue divisors
   matrixc&               shur_s,    // matrix S in Schur form
   matrixc&               shur_t,    // matrix T in Schur form
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixc&                vsl,       // matrix of left Schur vectors VSL
   matrixc&                vsr        // matrix of right Schur vectors VSR
   );
```

Computing for type matrix<complexf>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrixcf&               B,         // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,     // method to compute left and right vectors
   vectorcf&               alpha,     // vector of computed eigenvalues
   vectorcf&               beta,      // vector of eigenvalue divisors
   matrixcf&               shur_s,    // matrix S in Schur form
   matrixcf&               shur_t,    // matrix T in Schur form
   matrixcf&               vsl,       // matrix of left Schur vectors VSL
   matrixcf&               vsr        // matrix of right Schur vectors VSR
   );
```

Parameters

B

[in] The second matrix in the pair.

jobvs

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigenvalue divisors.

shur_s

[out] Matrix S, block upper triangular Schur matrix (Schur form for the input matrix).

shur_t

[out] Matrix T, block upper triangular Schur matrix (Schur form for the second matrix in the pair).

vsl

[out] Matrix of left Schur vectors VSL.

vsr

[out] Matrix of right Schur vectors VSR.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Computation depends on the *jobvs* parameter values.

© 2000-2025, MetaQuotes Ltd.

The second matrix in the pair must be the same size as the first (input) one.

Real (non-complex) matrices can have a complex solution. Therefore, the vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

EigenSolverShur

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000-2025, MetaQuotes Ltd.

Symmetric Matrices

Functions for computing eigenvalues and eigenvectors of symmetric or Hermitian matrices using the divide and conquer algorithm, making the process efficient and fast. These methods can be applied to matrices of different data types, including real and complex numbers.

Function	Action
EigenSymmetricDC	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the divide-and-conquer algorithm (lapack functions SYEVD, HEEVD).
EigenSymmetricQR	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the QR algorithm (lapack functions SYEV, HEEV).
EigenSymmetricRobust	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (lapack functions SYEVR, HEEVR).
EigenSymmetricBisect	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the bisection algorithm (lapack functions SYEVX, HEEVX).
EigenSymmetricDC2s	Compute all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal. If eigenvectors are desired, it uses a divide and conquer algorithm (lapack functions SYEVD_2STAGE, HEEVD_2STAGE).
EigenSymmetricQR2s	Compute all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal (lapack functions SYEV_2STAGE, HEEV_2STAGE).
EigenSymmetricRobust2s	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal then using the Multiple Relatively Robust Representations, MRRR algorithm (lapack functions SYEVR_2STAGE, HEEVR_2STAGE).
EigenSymmetricBisect2s	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction

© 2000-2025, MetaQuotes Ltd.

Function	Action
	to tridiagonal then using the bisection algorithm (lapack functions SYEVX_2STAGE, HEEVX_2STAGE).
EigenSymmetric2DC	Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.
EigenSymmetric2QR	Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.
EigenSymmetric2Bisect	Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricDC

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the divide-and-conquer algorithm (lapack functions SYEVD, HEEVD).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixcf&             eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated.

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricQR

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the QR algorithm (lapack functions SYEV, HEEV).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixcf&             eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated.

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricRobust

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (lapack functions SYEVR, HEEVR).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // Upper bound of the subset
   float                 abstol,            // absolute error tolerance
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf&              eigen_values,       // vector of computed eigenvectors
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

© 2000-2025, MetaQuotes Ltd.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricBisect

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the bisection algorithm (lapack functions SYEVX, HEEVX).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // Upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

V

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

© 2000-2025, MetaQuotes Ltd.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricDC2s

Compute all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal. If eigenvectors are desired, it uses a divide and conquer algorithm (lapack functions SYEVD_2STAGE, HEEVD_2STAGE).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixcf&             eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated. In the current OpenBLAS implementation, this value is not supported. Attempting to use it will result in error 4003 (ERR_INVALID_PARAMETER).

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated. Not available in this release.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricQR2s

Compute all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal (lapack functions SYEV_2STAGE, HEEV_2STAGE).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixcf&             eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated. In the current OpenBLAS implementation, this value is not supported. Attempting to use it will result in error 4003 (ERR_INVALID_PARAMETER).

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated. Not available in this release.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricRobust2s

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal then using the Multiple Relatively Robust Representations, MRRR algorithm (lapack functions SYEVR_2STAGE, HEEVR_2STAGE).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf&              eigen_values,       // vector of computed eigenvectors
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated. In the current OpenBLAS implementation, this value is not supported. Attempting to use it will result in error 4003 (ERR_INVALID_PARAMETER).

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

© 2000-2025, MetaQuotes Ltd.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated. Not available in this release.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricBisect2s

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal then using the bisection algorithm (lapack functions SYEVX_2STAGE, HEEVX_2STAGE).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf&              eigen_values,       // vector of computed eigenvectors
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

V

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated. In the current OpenBLAS implementation, this value is not supported. Attempting to use it will result in error 4003 (ERR_INVALID_PARAMETER).

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

© 2000-2025, MetaQuotes Ltd.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated. Not available in this release.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetric2DC

Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form

A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

Here A and B are assumed to be symmetric (Hermitian) and B is also positive definite. If eigenvectors are desired, it uses a divide-and-conquer algorithm (lapack functions SYGVD, HEGVD).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrix&               B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors,      // matrix of computed eigenvectors
   matrix&               triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixf&              B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors,      // matrix of computed eigenvectors
   matrixf&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixc&              B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors,      // matrix of computed eigenvectors
   matrixc&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixcf&             B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors,      // matrix of computed eigenvectors
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixcf&             triangular_factor   // triangular factor from the Cholesky fa
   );
```

Parameters

itype

[in] ENUM_EIGS2_TYPE enumeration value which specified the problem type to be solved : A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

B

[in] Second matrix B. Must be positive definite symmetric (or Hermitian conjugated) matrix.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

triangular_factor

[out] The triangular factor U or L from the Cholesky factorization B = U**T*U or B = L*L**T.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated). Second matrix B must be positive definite symmetric. If the input matrix and second matrix B are triangular, then both must be the same triangular, upper or lower.

ENUM_EIGS2_TYPE

An enumeration that specifies the problem type to be solved.

ID	Description
EIGS2TYPE_1	1: A*x = (lambda)*B*x
EIGS2TYPE_2	2: A*B*x = (lambda)*x
EIGS2TYPE_3	3: B*A*x = (lambda)*x

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

© 2000-2025, MetaQuotes Ltd.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetric2QR

Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form

A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

Here A and B are assumed to be symmetric (Hermitian) and B is also positive definite. This method uses QR algorithm (lapack functions SYGV, HEGV).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrix&               B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors,      // matrix of computed eigenvectors
   matrix&               triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixf&              B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors,      // matrix of computed eigenvectors
   matrixf&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixc&              B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors,      // matrix of computed eigenvectors
   matrixc&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixcf&             B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors,      // matrix of computed eigenvectors
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixcf&             triangular_factor   // triangular factor from the Cholesky fa
   );
```

Parameters

itype

[in] ENUM_EIGS2_TYPE enumeration value which specified the problem type to be solved : A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

B

[in] Second matrix B. Must be positive definite symmetric (or Hermitian conjugated) matrix.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

triangular_factor

[out] The triangular factor U or L from the Cholesky factorization B = U**T*U or B = L*L**T.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated). Second matrix B must be positive definite symmetric. If the input matrix and second matrix B are triangular, then both must be the same triangular, upper or lower.

ENUM_EIGS2_TYPE

An enumeration that specifies the problem type to be solved.

ID	Description
EIGS2TYPE_1	1: A*x = (lambda)*B*x
EIGS2TYPE_2	2: A*B*x = (lambda)*x
EIGS2TYPE_3	3: B*A*x = (lambda)*x

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

© 2000-2025, MetaQuotes Ltd.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetric2Bisect

Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form

A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

Here A and B are assumed to be symmetric (Hermitian) and B is also positive definite. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues. This method uses bisection algorithm (lapack functions SYGVX, HEGVX).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,              // subset of eigenvalues to compute
   double                lower,              // lower bound of the subset
   double                upper,              // Upper bound of the subset
   double                abstol,             // absolute error tolerance
   matrix&               B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors,      // matrix of computed eigenvectors
   matrix&               triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,              // subset of eigenvalues to compute
   float                 lower,              // lower bound of the subset
   float                 upper,              // upper bound of the subset
   float                 abstol,             // absolute error tolerance
   matrixf&              B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors,      // matrix of computed eigenvectors
   matrixf&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,              // subset of eigenvalues to compute
   double                lower,              // lower bound of the subset
   double                upper,              // Upper bound of the subset
   double                abstol,             // absolute error tolerance
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixc&              B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors,      // matrix of computed eigenvectors
   matrixc&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,              // subset of eigenvalues to compute
   float                 lower,              // lower bound of the subset
   float                 upper,              // Upper bound of the subset
   float                 abstol,             // absolute error tolerance
   matrixcf&             B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors,      // matrix of computed eigenvectors
   matrixcf&             triangular_factor   // triangular factor from the Cholesky fa
   );
```

Parameters

itype

[in] ENUM_EIGS2_TYPE enumeration value which specified the problem type to be solved : A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

© 2000-2025, MetaQuotes Ltd.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

B

[in] Second matrix B. Must be positive definite symmetric (or Hermitian conjugated) matrix.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

triangular_factor

[out] The triangular factor U or L from the Cholesky factorization B = U**T*U or B = L*L**T.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated). Second matrix B must be positive definite symmetric. If the input matrix and second matrix B are triangular, then both must be the same triangular, upper or lower.

ENUM_EIGS2_TYPE

An enumeration that specifies the problem type to be solved.

ID	Description
EIGS2TYPE_1	1: A*x = (lambda)*B*x
EIGS2TYPE_2	2: A*B*x = (lambda)*x
EIGS2TYPE_3	3: B*A*x = (lambda)*x

© 2000-2025, MetaQuotes Ltd.

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000-2025, MetaQuotes Ltd.

Tridiagonal Matrices

Functions for computing eigenvalues and eigenvectors of symmetric tridiagonal matrices using various algorithms. Each function implements a specific solution method and supports matrix types double and float.

Common Parameters:

- jobv — Determines whether to compute eigenvectors (EIGVALUES_V) or only eigenvalues (EIGVALUES_N).
- range — Specifies the range of computed eigenvalues (BLASRANGE_A, BLASRANGE_V, BLASRANGE_I).
- lower and upper — Lower and upper bounds for computing a subset of the spectrum.
- abstol — Absolute error tolerance.

All functions operate on symmetric tridiagonal matrices and allow selecting the most suitable algorithm depending on performance and accuracy requirements.

Function	Action
EigenTridiagonalDC	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the divide-and-conquer algorithm (lapack function STEVD).
EigenTridiagonalQR	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm (lapack function STEV).
EigenTridiagonalRobust	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the Multiple Relatively Robust Representations, MRRR algorithm (lapack function STEVR).
EigenTridiagonalBisect	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the bisection algorithm (lapack function STEVX).
EigenTridiagonalQL	Compute all eigenvalues of a symmetric tridiagonal matrix using the Pal-Walker-Kahan variant of the QL or QR algorithm (lapack function STERF).
EigenTridiagonalDCQ	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the divide-and-conquer algorithm (lapack function STEDC).
EigenTridiagonalQRQ	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm (lapack function STEQR).
EigenTridiagonalPosDefQ	Compute eigenvalues and eigenvectors of a symmetric positive definite (положительно

© 2000-2025, MetaQuotes Ltd.

Function	Action
	определённая) tridiagonal matrix using the QR algorithm (lapack function PTEQR).

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalDC

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the divide-and-conquer algorithm (lapack function STEVD).

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalDC(
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalDC(
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated.

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

© 2000-2025, MetaQuotes Ltd.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalQR

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm (lapack function STEV).

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalQR(
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalQR(
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated.

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

© 2000-2025, MetaQuotes Ltd.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalRobust

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the Multiple Relatively Robust Representations, MRRR algorithm (lapack function STEVR).

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

© 2000-2025, MetaQuotes Ltd.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.

© 2000-2025, MetaQuotes Ltd.

ID	Description
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalBisect

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the bisection algorithm (lapack function STEVX).

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

© 2000-2025, MetaQuotes Ltd.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

V

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.

© 2000-2025, MetaQuotes Ltd.

ID	Description
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalQL

Compute all eigenvalues of a symmetric tridiagonal matrix using the Pal-Walker-Kahan variant of the QL or QR algorithm (lapack function STERF).

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalQL(
   vector&               eigen_values        // vector of computed eigenvalues
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalQL(
   vectorf&              eigen_values        // vector of computed eigenvalues
   );
```

Parameters

eigen_values

[out] Vector of eigenvalues.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input must be a symmetric matrix in the tridiagonal form.

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalDCQ

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the divide-and-conquer algorithm (lapack function STEDC). Unlike EigenTridiagonalDC, this method can be used to compute the eigenvectors of the original symmetric matrix. A symmetric matrix can be reduced to tridiagonal form using the ReduceSymmetricToTridiagonal method. The orthogonal matrix Q obtained from this transformation is then used to compute the eigenvectors of the original symmetric matrix.

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrix&               Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixf&              Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixc&              Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixcf&             Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

compv

[in] ENUM_EIGTRIDIAG_Z enumeration value which determines the method for computing eigenvectors.

Q

[in] Orthogonal matrix Q produced by method ReflectTridiagonalToQ.

© 2000-2025, MetaQuotes Ltd.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the compv parameter.

When compv = EIGCOMPZ_N, compute eigenvalues only, eigenvectors are not calculated.

If EIGCOMPZ_V is set, eigenvalues are computed and eigenvectors of original symmetric matrix are calculated also.

If EIGCOMPZ_I is set, eigenvalues are computed and eigenvectors of tridiagonal matrix are calculated also.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIGTRIDIAG_Z

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGCOMPZ_N	'N': Compute eigenvalues only
EIGCOMPZ_V	'V': Compute eigenvectors of original symmetric matrix also
EIGCOMPZ_I	'I': Compute eigenvectors of tridiagonal matrix also

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalQRQ

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm (lapack function STEQR). Unlike EigenTridiagonalQR, this method can be used to compute the eigenvectors of the original symmetric matrix. A symmetric matrix can be reduced to tridiagonal form using the ReduceSymmetricToTridiagonal method. The orthogonal matrix Q obtained from this transformation is then used to compute the eigenvectors of the original symmetric matrix.

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrix&               Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixf&              Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixc&              Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixcf&             Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

compv

[in] ENUM_EIGTRIDIAG_Z enumeration value which determines the method for computing eigenvectors.

Q

[in] Orthogonal matrix Q produced by method ReflectTridiagonalToQ.

© 2000-2025, MetaQuotes Ltd.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the compv parameter.

When compv = EIGCOMPZ_N, compute eigenvalues only, eigenvectors are not calculated.

If EIGCOMPZ_V is set, eigenvalues are computed and eigenvectors of original symmetric matrix are calculated also.

If EIGCOMPZ_I is set, eigenvalues are computed and eigenvectors of tridiagonal matrix are calculated also.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIGTRIDIAG_Z

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGCOMPZ_N	'N': Compute eigenvalues only
EIGCOMPZ_V	'V': Compute eigenvectors of original symmetric matrix also
EIGCOMPZ_I	'I': Compute eigenvectors of tridiagonal matrix also

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalPosDefQ

Compute eigenvalues and eigenvectors of a symmetric positive definite tridiagonal matrix using the QR algorithm (lapack function PTEQR). This method can be used to compute the eigenvectors of both the input tridiagonal matrix and the original positive definite symmetric matrix. This is achieved using the orthogonal matrix Q obtained during the reduction to tridiagonal form.

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrix&               Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixf&              Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixc&              Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixcf&             Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

compv

[in] ENUM_EIGTRIDIAG_Z enumeration value which determines the method for computing eigenvectors.

Q

[in] Orthogonal matrix Q produced by method ReflectTridiagonalToQ.

eigen_values

© 2000-2025, MetaQuotes Ltd.

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the compv parameter.

When compv = EIGCOMPZ_N, compute eigenvalues only, eigenvectors are not calculated.

If EIGCOMPZ_V is set, eigenvalues are computed and eigenvectors of original symmetric matrix are calculated also.

If EIGCOMPZ_I is set, eigenvalues are computed and eigenvectors of tridiagonal matrix are calculated also.

The input must be a symmetric positive definite tridiagonal matrix.

ENUM_EIGTRIDIAG_Z

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGCOMPZ_N	'N': Compute eigenvalues only
EIGCOMPZ_V	'V': Compute eigenvectors of original symmetric matrix also
EIGCOMPZ_I	'I': Compute eigenvectors of tridiagonal matrix also

© 2000-2025, MetaQuotes Ltd.

Orthogonal Factorizations

OpenBLAS предоставляет ряд процедур для разложения общей прямоугольной матрицы A размера m на n в произведение ортогональной (в комплексном случае — унитарной) и треугольной (или, возможно, трапециевидной) матриц.

Реальная матрица Q называется ортогональной, если $Q^T Q = I$; комплексная матрица Q называется унитарной, если $Q^H Q = I$. Ортогональные и унитарные матрицы обладают важным свойством — они сохраняют евклидову норму вектора:

$||x||_2 = ||Qx||_2$, если Q ортогональна или унитарна

Благодаря этому такие матрицы способствуют численной устойчивости, поскольку не усиливают ошибки округления.

Ортогональные разложения широко применяются при решении задач наименьших квадратов. Их также можно использовать для выполнения предварительных шагов при решении задач на собственные значения или сингулярные значения.

Функция	Выполняемое действие
FactorizationQR	Вычисляет QR-разложение общей матрицы размера m на n: A = Q * R. LAPACK-функция GEQRF.
FactorizationQRNonNeg	Вычисляет QR-разложение общей матрицы размера m на n: A = Q * R, где R — верхнетреугольная матрица с неотрицательными элементами на диагонали. LATPACK-функция GEQRFP.
FactorizationQRPivot	Вычисляет QR-разложение общей матрицы размера m на n с перестановкой столбцов: A * P = Q * R. LAPACK-функция GEQP3.
FactorizationLQ	Computes the LQ factorization of a general m-by-n matrix: A = L * Q. Lapack function GELQF.
FactorizationQL	Computes the QL factorization of a general m-by-n matrix: A = Q * L. Lapack function GEQLF.
FactorizationRQ	Computes the RQ factorization of a general m-by-n matrix: A = R * Q. Lapack function GERQF.

© 2000-2025, MetaQuotes Ltd.

FactorizationQR

Computes the QR factorization of a general m-by-n matrix: A = Q * R. Lapack function GEQRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationQR(
   bool            reduced,      // calculation mode reduced or complete
   matrix&         Q,            // orthogonal matrix Q
   matrix&         R             // upper triangular matrix R
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationQR(
   bool            reduced,      // calculation mode reduced or complete
   matrixf&        Q,            // orthogonal matrix Q
   matrixf&        R             // upper triangular matrix R
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationQR(
   bool            reduced,      // calculation mode reduced or complete
   matrixc&        Q,            // unitary matrix Q
   matrixc&        R             // upper triangular matrix R
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationQR(
   bool            reduced,      // calculation mode reduced or complete
   matrixcf&       Q,            // unitary matrix Q
   matrixcf&       R             // upper triangular matrix R
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices Q, R calculated with dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices Q, R with dimensions (M,M), (M,N).

Q

[out] Orthogonal or unitary matrix Q.

R

[out] Upper triangular matrix R.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

Note

If *reduced* is true

If m >= n, matrix Q is of m-by-n sizes, matrix R is of n-by-n sizes.

If m < n, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

If *reduced* is false, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

© 2000-2025, MetaQuotes Ltd.

FactorizationQRNonNeg

Computes the QR factorization of a general m-by-n matrix: A = Q * R. R is an upper triangular matrix with nonnegative diagonal entries. Lapack function GEQRFP.

Computing for type matrix<double>

```
bool  matrix::FactorizationQRNonNeg(
   bool             reduced,       // calculation mode reduced or complete
   matrix&          Q,             // orthogonal matrix Q
   matrix&          R              // upper triangular matrix R
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationQRNonNeg(
   bool             reduced,       // calculation mode reduced or complete
   matrixf&         Q,             // orthogonal matrix Q
   matrixf&         R              // upper triangular matrix R
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationQRNonNeg(
   bool             reduced,       // calculation mode reduced or complete
   matrixc&         Q,             // unitary matrix Q
   matrixc&         R              // upper triangular matrix R
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationQRNonNeg(
   bool             reduced,       // calculation mode reduced or complete
   matrixcf&        Q,             // unitary matrix Q
   matrixcf&        R              // upper triangular matrix R
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices Q, R calculated with dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices Q, R with dimensions (M,M), (M,N).

Q

[out] Orthogonal or unitary matrix Q.

R

[out] Upper triangular matrix R.

Return Value

© 2000-2025, MetaQuotes Ltd.

Return true if successful, otherwise false in case of an error.

Note

If *reduced* is true

If m >= n, matrix Q is of m-by-n sizes, matrix R is of n-by-n sizes.

If m < n, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

If *reduced* is false, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

© 2000-2025, MetaQuotes Ltd.

FactorizationQRPivot

Computes the QR factorization of a general m-by-n matrix with column pivoting: A * P = Q * R. Lapack function GEQP3.

Computing for type matrix<double>

```
bool  matrix::FactorizationQRPivot(
   bool             reduced,      // calculation mode reduced or complete
   long[]&          jpvt,         // array with predefined permutations
   matrix&          Q,            // orthogonal matrix Q
   matrix&          R,            // upper triangular matrix R
   matrix&          P             // permutation matrix
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationQRPivot(
   bool             reduced,      // calculation mode reduced or complete
   long[]&          jpvt,         // array with predefined permutations
   matrixf&         Q,            // orthogonal matrix Q
   matrixf&         R,            // upper triangular matrix R
   matrixf&         P             // permutation matrix
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationQRPivot(
   bool             reduced,      // calculation mode reduced or complete
   long[]&          jpvt,         // array with predefined permutations
   matrixc&         Q,            // unitary matrix Q
   matrixc&         R,            // upper triangular matrix R
   matrixc&         P             // permutation matrix
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationQRPivot(
   bool             reduced,      // calculation mode reduced or complete
   long[]&          jpvt,         // array with predefined permutations
   matrixcf&        Q,            // unitary matrix Q
   matrixcf&        R,            // upper triangular matrix R
   matrixcf&        P             // permutation matrix
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices Q, R calculated with dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices Q, R with dimensions (M,M), (M,N).

© 2000-2025, MetaQuotes Ltd.

jpvt

[in] Integer array of dimension n. if jpvt(i) ≠ 0, the i-th column of A is moved to the beginning of AP before the computation, and fixed in place during the computation. If jpvt(i) = 0, the i-th column of A is a free column (that is, it may be interchanged during the computation with any other free column). If array has zero size (or not initialized), then all the columns of A assumed to be free.

Q

[out] Orthogonal or unitary matrix Q.

R

[out] Upper triangular matrix R.

P

[out] Permutation matrix P of n-by-n sizes.

Return Value

Return true if successful, otherwise false in case of an error.

Note

If *reduced* is true

If m >= n, matrix Q is of m-by-n sizes, matrix R is of n-by-n sizes.

If m < n, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

If *reduced* is false, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

© 2000-2025, MetaQuotes Ltd.

FactorizationLQ

Computes the LQ factorization of a general m-by-n matrix: A = L * Q. Lapack function GELQF.

Computing for type matrix<double>

```
bool  matrix::FactorizationLQ(
   bool            reduced,      // calculation mode reduced or complete
   matrix&         L,            // lower triangular matrix L
   matrix&         Q             // orthogonal matrix Q
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationLQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixf&        L,            // lower triangular matrix L
   matrixf&        Q             // orthogonal matrix Q
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationLQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixc&        L,            // lower triangular matrix L
   matrixc&        Q             // unitary matrix Q
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationLQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixcf&       L,            // lower triangular matrix L
   matrixcf&       Q             // unitary matrix Q
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices L, Q calculated with reduced dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices L, Q with dimensions (M,N), (N,N).

L

[out] Lower triangular matrix L.

Q

[out] Orthogonal or unitary matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

Note

If *reduced* is true

If m <= n, matrix L is of m-by-m sizes, matrix Q is of m-by-n sizes.

If m > n, matrix L is of m-by-n sizes, matrix Q is of n-by-n sizes.

If *reduced* is false, matrix L is of m-by-n sizes, matrix Q is of n-by-n sizes.

© 2000-2025, MetaQuotes Ltd.

FactorizationQL

Computes the QL factorization of a general m-by-n matrix: A = Q * L. Lapack function GEQLF.

Computing for type matrix<double>

```
bool  matrix::FactorizationQL(
   bool            reduced,      // calculation mode reduced or complete
   matrix&         Q,            // orthogonal matrix Q
   matrix&         L             // lower triangular matrix L
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationQL(
   bool            reduced,      // calculation mode reduced or complete
   matrixf&        Q,            // orthogonal matrix Q
   matrixf&        L             // lower triangular matrix L
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationQL(
   bool            reduced,      // calculation mode reduced or complete
   matrixc&        Q,            // unitary matrix Q
   matrixc&        L             // lower triangular matrix L
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationQL(
   bool            reduced,      // calculation mode reduced or complete
   matrixcf&       Q,            // unitary matrix Q
   matrixcf&       L             // lower triangular matrix L
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices Q, L calculated with reduced dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices Q, L with dimensions (M,M), (M,N).

Q

[out] Orthogonal or unitary matrix Q.

L

[out] Lower triangular matrix L.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

Note

If *reduced* is true

If m >= n, matrix Q is of m-by-n sizes, matrix L is of n-by-n sizes.

If m < n, matrix Q is of m-by-m sizes, matrix L is of m-by-n sizes.

If *reduced* is false, matrix Q is of m-by-m sizes, matrix L is of m-by-n sizes.

© 2000-2025, MetaQuotes Ltd.

FactorizationRQ

Computes the RQ factorization of a general m-by-n matrix: A = R * Q. Lapack function GERQF.

Computing for type matrix<double>

```
bool  matrix::FactorizationRQ(
   bool            reduced,      // calculation mode reduced or complete
   matrix&         R,            // upper triangular matrix R
   matrix&         Q             // orthogonal matrix Q
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationRQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixf&        R,            // upper triangular matrix R
   matrixf&        Q             // orthogonal matrix Q
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationRQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixc&        R,            // upper triangular matrix R
   matrixc&        Q             // unitary matrix Q
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationRQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixcf&       R,            // upper triangular matrix R
   matrixcf&       Q             // unitary matrix Q
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices R, Q calculated with reduced dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices L, Q with dimensions (M,N), (N,N).

R

[out] Upper triangular matrix R.

Q

[out] Orthogonal or unitary matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

Note

If *reduced* is true

If m <= n, matrix R is of m-by-m sizes, matrix Q is of m-by-n sizes.

If m > n, matrix R is of m-by-n sizes, matrix Q is of n-by-n sizes.

If *reduced* is false, matrix R is of m-by-n sizes, matrix Q is of n-by-n sizes.

© 2000-2025, MetaQuotes Ltd.

Linear Equations

This section provides functions for solving systems of linear equations of the form A·X = B, where A is a square coefficient matrix, B is the right-hand side, and X is the solution matrix or vector. The methods support various data types (double, float, complex, complexf) and are based on LAPACK functions to ensure high precision and performance.

The functions are categorized based on the type of the coefficient matrix A. Each function is overloaded to support both matrix and vector forms for the right-hand side (B) and the solution (X).

Function	Action
LinearEquationsSolution	Computes the solution to the system of linear equations with a square coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function GESV.
LinearEquationsSolutionTriangular	Computes the solution to the system of linear equations with a square triangular coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function TRTRS.
LinearEquationsSolutionSy	Computes the solution to the system of linear equations with a symmetric or Hermitian conjugated matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric or Hermitian conjugated matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack functions SYSV, HESV.
LinearEquationsSolutionComplexSy	Computes the solution to the system of linear equations with a complex symmetric (not Hermitian conjugated!) matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n complex symmetric matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function SYSV.
LinearEquationsSolutionSyPD	Computes the solution to the system of linear equations with a symmetric or Hermitian conjugated positive-definite matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric or Hermitian conjugated positive-definite matrix, the columns of matrix B are individual right-hand sides,

© 2000-2025, MetaQuotes Ltd.

Function	Action
	and the columns of X are the corresponding solutions. Lapack function POSV.
LinearEquationsSolutionGeTrid	Computes the solution to the system of linear equations with a general (nonsymmetric) tridiagonal coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general tridiagonal matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function GTSV.
LinearEquationsSolutionSyTridPD	Computes the solution to the system of linear equations with a symmetric tridiagonal positive-definite coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric tridiagonal matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function PTSV.

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolution

Computes the solution to the system of linear equations with a square coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function GESV.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolution(
   matrix&          B,             // right hand side matrix B
   matrix&          X              // solution matrix X
   );

bool  matrix::LinearEquationsSolution(
   vector&          B,             // right hand side vector B
   vector&          X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolution(
   matrixf&         B,             // right hand side matrix B
   matrixf&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolution(
   vectorf&          B,             // right hand side vector B
   vectorf&          X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolution(
   matrixc&         B,             // right hand side matrix B
   matrixc&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolution(
   vectorc&         B,             // right hand side vector B
   vectorc&         X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolution(
   matrixcf&        B,             // right hand side matrix B
   matrixcf&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolution(
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorcf&       B,            // right hand side vector B
   vectorcf&       X             // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionTriangular

Computes the solution to the system of linear equations with a square triangular coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function TRTRS.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrix&             B,            // right hand side matrix B
   matrix&             X             // solution matrix X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vector&             B,            // right hand side vector B
   vector&             X             // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixf&            B,            // right hand side matrix B
   matrixf&            X             // solution matrix X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorf&            B,            // right hand side vector B
   vectorf&            X             // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixc&            B,            // right hand side matrix B
   matrixc&            X             // solution matrix X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorc&            B,            // right hand side vector B
   vectorc&            X             // solution vector X
   );
```

Computing for type matrix<complexf>

© 2000-2025, MetaQuotes Ltd.

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixcf&           B,            // right hand side matrix B
   matrixcf&           X             // solution matrix X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorcf&           B,            // right hand side vector B
   vectorcf&           X             // solution vector X
   );
```

Parameters

trans

[in] ENUM_EQUATIONS_FORM enumeration value which specifies the form of the system of equations.

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

ENUM_EQUATIONS_FORM

An enumeration defining which form of the equations' system calculated.

ID	Description
EQUATIONSFORM_N	'N': A * X = B (No transpose)
EQUATIONSFORM_T	'T': A**T * X = B (Transpose)
EQUATIONSFORM_C	'C': A**H * X = B (Conjugate transpose)

In case of real matrices the value EQUATIONSFORM_C assumed as Transpose.

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionSy

Computes the solution to the system of linear equations with a symmetric or Hermitian conjugated matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric or unitary matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack functions SYSV, HESV.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolutionSy(
   matrix&          B,             // right hand side matrix B
   matrix&          X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSy(
   vector&          B,             // right hand side vector B
   vector&          X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolutionSy(
   matrixf&         B,             // right hand side matrix B
   matrixf&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSy(
   vectorf&          B,             // right hand side vector B
   vectorf&          X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionSy(
   matrixc&         B,             // right hand side matrix B
   matrixc&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSy(
   vectorc&         B,             // right hand side vector B
   vectorc&         X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolutionSy(
   matrixcf&        B,             // right hand side matrix B
   matrixcf&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSy(
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorcf&       B,              // right hand side vector B
   vectorcf&       X               // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

The input can be a symmetric (Hermitian conjugated), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionComplexSy

Computes the solution to the system of linear equations with a complex symmetric (not Hermitian conjugated!) matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n complex symmetric matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function SYSV.

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionComplexSy(
   matrixc&        B,             // right hand side matrix B
   matrixc&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionComplexSy(
   vectorc&        B,             // right hand side vector B
   vectorc&        X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolutionComplexSy(
   matrixcf&       B,             // right hand side matrix B
   matrixcf&       X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionComplexSy(
   vectorcf&       B,             // right hand side vector B
   vectorcf&       X              // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

The input can be a complex symmetric, upper triangular or lower triangular matrix. Triangular matrices are assumed to be complex symmetric.

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionSyPD

Computes the solution to the system of linear equations with a symmetric or Hermitian conjugated positive-definite matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric or unitary positive-definite matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function POSV.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrix&         B,             // right hand side matrix B
   matrix&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyPD(
   vector&         B,             // right hand side vector B
   vector&         X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrixf&        B,             // right hand side matrix B
   matrixf&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyPD(
   vectorf&         B,             // right hand side vector B
   vectorf&         X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrixc&        B,             // right hand side matrix B
   matrixc&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyPD(
   vectorc&        B,             // right hand side vector B
   vectorc&        X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrixcf&       B,             // right hand side matrix B
   matrixcf&       X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyPD(
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorcf&       B,              // right hand side vector B
   vectorcf&       X               // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

The input can be a symmetric (Hermitian conjugated), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionGeTrid

Computes the solution to the system of linear equations with a general (nonsymmetric) tridiagonal coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general tridiagonal matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function GTSV.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolutionGeTrid(
   matrix&         B,             // right hand side matrix B
   matrix&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionGeTrid(
   vector&         B,             // right hand side vector B
   vector&         X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolutionGeTrid(
   matrixf&        B,             // right hand side matrix B
   matrixf&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionGeTrid(
   vectorf&         B,             // right hand side vector B
   vectorf&         X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionGeTrid(
   matrixc&        B,             // right hand side matrix B
   matrixc&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionGeTrid(
   vectorc&        B,             // right hand side vector B
   vectorc&        X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolutionGeTrid(
   matrixcf&       B,             // right hand side matrix B
   matrixcf&       X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionGeTrid(
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorcf&       B,             // right hand side vector B
   vectorcf&       X              // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionSyTridPD

Computes the solution to the system of linear equations with a symmetric tridiagonal positive-definite coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric tridiagonal matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. Lapack function PTSV.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolutionSyTridPD(
   matrix&         B,             // right hand side matrix B
   matrix&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyTridPD(
   vector&         B,             // right hand side vector B
   vector&         X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolutionSyTridPD(
   matrixf&        B,             // right hand side matrix B
   matrixf&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyTridPD(
   vectorf&         B,             // right hand side vector B
   vectorf&         X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionSyTridPD(
   matrixc&        B,             // right hand side matrix B
   matrixc&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyTridPD(
   vectorc&        B,             // right hand side vector B
   vectorc&        X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolutionSyTridPD(
   matrixcf&       B,             // right hand side matrix B
   matrixcf&       X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyTridPD(
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorcf&       B,            // right hand side vector B
   vectorcf&       X             // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000-2025, MetaQuotes Ltd.

Factorizations

The Factorizations section contains functions for performing various types of matrix factorizations used in numerical solutions of linear systems, stability analysis, and other linear algebra tasks. These factorizations transform the original matrix into simpler forms, making subsequent computations more efficient. All functions are implemented using LAPACK routines and support the types double, float, complex, and complexf.

The functions in this section are used for:

- Preprocessing matrices when solving systems of linear equations;
- Computing determinants, ranks, and matrix inverses;
- Assessing the stability of numerical methods;
- Solving problems in spectral theory and optimization methods.

Matrix factorization is a critical step in many linear algebra algorithms, and this section provides access to the most efficient and well-established factorization techniques.

Function	Action
FactorizationPLU	Computes an LU factorization of a general M-by-N matrix A using partial pivoting with row interchanges. The factorization has the form A = P * L * U. Lapack function GETRF.
FactorizationPLUGeTrid	Computes an LU factorization of a general (non-symmetric) tridiagonal N-by-N matrix A using elimination with partial pivoting and row interchanges. The factorization has the form A = P * L * U. Lapack function GTTRF.
FactorizationLDL	Computes the factorization of a real symmetric or complex Hermitian matrix A using the Bunch-Kaufman diagonal pivoting method. Lapack functions SYTRF, HETRF.
FactorizationLDLSyTridPD	Forms the factorization of a symmetric positive-definite or, for complex data, Hermitian positive-definite tridiagonal matrix A. Lapack function PTTRF.
FactorizationCholesky	Computes the factorization of a real symmetric or complex Hermitian positive-definite matrix A. Lapack function POTRF.
FactorizationCholeskySyPS	Computes the Cholesky factorization with complete pivoting of a real symmetric (complex Hermitian) positive semidefinite N-by-N matrix. Lapack function PSTRF.

© 2000-2025, MetaQuotes Ltd.

FactorizationPLU

Computes an LU factorization of a general M-by-N matrix A using partial pivoting with row interchanges. The factorization has the form

A = P * L * U

where P is a permutation matrix, L is lower triangular with unit diagonal elements (lower trapezoidal if m > n), and U is upper triangular (upper trapezoidal if m < n). Lapack function GETRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationPLU(
   matrix&          P,             // permutation matrix P
   matrix&          L,             // lower triangular matrix L
   matrix&          U              // upper triangular matrix U
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationPLU(
   matrixf&         P,             // permutation matrix P
   matrixf&         L,             // lower triangular matrix L
   matrixf&         U              // upper triangular matrix U
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationPLU(
   matrixc&         P,             // permutation matrix P
   matrixc&         L,             // lower triangular matrix L
   matrixc&         U              // upper triangular matrix U
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationPLU(
   matrixcf&        P,             // permutation matrix P
   matrixcf&        L,             // lower triangular matrix L
   matrixcf&        U              // upper triangular matrix U
   );
```

Parameters

P

[out] Permutation matrix P.

L

[out] Lower triangular matrix L with unit diagonal elements.

U

[out] Upper triangular matrix U.

© 2000-2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

FactorizationPLUGeTrid

Computes an LU factorization of a general (non-symmetric) tridiagonal N-by-N matrix A using elimination with partial pivoting and row interchanges. The factorization has the form

A = P * L * U

where P is a permutation matrix, L is lower triangular with unit diagonal elements, and U is upper triangular. Lapack function GTTRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationPLUGeTrid(
   matrix&          P,             // permutation matrix P
   matrix&          L,             // lower triangular matrix L
   matrix&          U              // upper triangular matrix U
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationPLUGeTrid(
   matrixf&         P,             // permutation matrix P
   matrixf&         L,             // lower triangular matrix L
   matrixf&         U              // upper triangular matrix U
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationPLUGeTrid(
   matrixc&         P,             // permutation matrix P
   matrixc&         L,             // lower triangular matrix L
   matrixc&         U              // upper triangular matrix U
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationPLUGeTrid(
   matrixcf&        P,             // permutation matrix P
   matrixcf&        L,             // lower triangular matrix L
   matrixcf&        U              // upper triangular matrix U
   );
```

Parameters

P

[out] Permutation matrix P.

L

[out] Lower triangular matrix L with unit diagonal elements.

U

[out] Upper triangular matrix U.

© 2000-2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

FactorizationLDL

Computes the factorization of a real symmetric or complex Hermitian matrix A using the Bunch-Kaufman diagonal pivoting method. The form of the factorization is:

A = L * D * L**T in case of lower triangular or symmetric matrix A

or

A = U**T * D * U in case of upper triangular matrix A

where L is lower triangular with unit diagonal elements, U is upper triangular with unit diagonal elements. D is a symmetric block-diagonal matrix with 1-by-1 and 2-by-2 diagonal blocks. Lapack functions SYTRF, HETRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationLDL(
   matrix&          L,             // lower or upper triangular matrix
   matrix&          D              // diagonal matrix D
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationLDL(
   matrixf&         L,             // lower or upper triangular matrix
   matrixf&         D              // diagonal matrix D
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationLDL(
   matrixc&         L,             // lower or upper triangular matrix
   matrixc&         D              // diagonal matrix D
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationLDL(
   matrixcf&        L,             // lower or upper triangular matrix
   matrixcf&        D              // diagonal matrix D
   );
```

Parameters

L

[out] Lower or upper triangular matrix with unit diagonal elements.

D

[out] Symmetric block-diagonal matrix D.

Return Value

© 2000-2025, MetaQuotes Ltd.

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

© 2000-2025, MetaQuotes Ltd.

FactorizationLDLSyTridPD

Forms the factorization of a symmetric positive-definite or, for complex data, Hermitian positive-definite tridiagonal matrix A:

A = L * D * L**T in case of lower triangular or symmetric matrix A

or

A = U**T * D * U in case of upper triangular matrix A

where L is lower triangular with unit diagonal elements, U is upper triangular with unit diagonal elements. D is diagonal matrix. Lapack function PTTRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrix&         L,            // lower or upper triangular matrix
   matrix&         D             // diagonal matrix D
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrixf&        L,            // lower or upper triangular matrix
   matrixf&        D             // diagonal matrix D
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrixc&        L,            // lower or upper triangular matrix
   matrixc&        D             // diagonal matrix D
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrixcf&       L,            // lower or upper triangular matrix
   matrixcf&       D             // diagonal matrix D
   );
```

Parameters

L

[out] Lower or upper triangular matrix with unit diagonal elements.

D

[out] Diagonal matrix D.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

© 2000-2025, MetaQuotes Ltd.

FactorizationCholesky

Computes the factorization of a real symmetric or complex Hermitian positive-definite matrix A. The factorization has the form:

A = L * L**T in case of lower triangular or symmetric matrix A

or

A = U**T * U in case of upper triangular matrix A

where L is lower triangular, U is upper triangular. Lapack function POTRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationCholesky(
   matrix&          L              // lower or upper triangular matrix
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationCholesky(
   matrixf&         L              // lower or upper triangular matrix
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationCholesky(
   matrixc&         L              // lower or upper triangular matrix
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationCholesky(
   matrixcf&        L              // lower or upper triangular matrix
   );
```

Parameters

L

[out] Lower or upper triangular matrix.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

© 2000-2025, MetaQuotes Ltd.

FactorizationCholeskySyPS

Computes the Cholesky factorization with complete pivoting of a real symmetric (complex Hermitian) positive semidefinite N-by-N matrix. The form of the factorization is:

P**T * A * P = L * L**T in case of lower triangular or symmetric matrix A

or

P**T * A * P = U**T * U in case of upper triangular matrix A

where P is a permutation matrix, L is lower triangular, U is upper triangular. Lapack function PSTRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationCholeskySyPS(
   double          tol           // tolerance
   matrix&         P,            // permutation matrix P
   matrix&         L             // lower or upper triangular matrix
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationCholeskySyPS(
   float           tol           // tolerance
   matrixf&        P,            // permutation matrix P
   matrixf&        L             // lower or upper triangular matrix
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationCholeskySyPS(
   double          tol           // tolerance
   matrixc&        P,            // permutation matrix P
   matrixc&        L             // lower or upper triangular matrix
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationCholeskySyPS(
   float           tol           // tolerance
   matrixcf&       P,            // permutation matrix P
   matrixcf&       L             // lower or upper triangular matrix
   );
```

Parameters

tol

[in] User defined tolerance. If tol < 0, then n*ε*max(A[k,k]), where ε is the machine precision, will be used. The algorithm terminates at the (k-1)st step, if the pivot <=tol.

P

[out] Permutation matrix P.

© 2000-2025, MetaQuotes Ltd.

L

[out] Lower or upper triangular matrix.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

© 2000-2025, MetaQuotes Ltd.

Dynamic Mode Decomposition

The Dynamic Mode Decomposition (DMD) functions are designed to compute an evolution operator from data snapshots, enabling efficient analysis of linear dynamics in extended phase spaces. The methods provide memory-efficient representations of evolution operators, with DynamicModeDecompositionQR incorporating QR-based data compression for improved computational efficiency.

Function	Action
DynamicModeDecomposition	Compute the Dynamic Mode Decomposition (DMD) for a pair of data snapshot matrices. Lapack function GEDMD.
DynamicModeDecompositionQR	Compute the Dynamic Mode Decomposition (DMD) for a pair of data snapshot matrices, using a QR factorization based compression of the data. Lapack function GEDMDQ.

The Dynamic mode decomposition (DMD) method is an algorithm for searching for an evolution operator (inverse operator problem solutions) in a finite-dimensional problem solution space (numerical or experimentally obtained) in a set of solutions (slices, "snapshots") in some consecutive moments of time. Expansion of the phase space due to the use of a nonlinear basis (relative to the variables of the problem) allows us to construct a global linear operator describing a linear evolution in the extended "rectifying space" (the Coopman operator) and the Perron-Frobenius operator that is its adjoint one. The DMD method is equivalent to a compressed representation of a linear evolution operator in the form of a product of rectangular matrices, which provides significant savings in the required memory during calculations.

© 2000-2025, MetaQuotes Ltd.

DynamicModeDecomposition

Compute the Dynamic Mode Decomposition(по-русски "разложение по динамическим модам") (DMD) for a pair of data snapshot matrices. Lapack function GEDMD. For the input matrices X and Y such that Y = A*X with an unaccessible matrix A, GEDMD computes a certain number of Ritz pairs of A using the standard Rayleigh-Ritz extraction from a subspace of range(X) that is determined using the leading left singular vectors of X. Optionally, GEDMD returns the residuals of the computed Ritz pairs, the information needed for a refinement of the Ritz vectors, or the eigenvectors of the Exact DMD. Both matrices must be the same size.

Computing for type matrix<double>

```
bool  matrix::DynamicModeDecomposition(
   matrix&               B,                  // second snapshot matrix in the pai
   ENUM_DMD_SCALE        jobs,               // determines whether the initial da
   ENUM_DMD_EIGV         jobz,               // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,               // determines whether to compute the
   ENUM_DMD_REFINE       jobf,               // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,             // allows for a selection of the SVD
   long                  nrnk,               // determines the mode how to comput
   double                tol,                // the tolerance for truncating smal
   vectorc&              eigen_values,       // vector of eigenvalues (Ritz value
   matrix&               left_vectors,       // matrix of left singular vectors
   matrix&               Z                   // matrix of Ritz vectors
   vector&               residuals,          // residuals for the Ritz pairs
   matrix&               res_vectors,        // residual vectors for the Ritz pai
   matrix&               B,                  // computed vectors
   matrix&               W,                  // eigenvectors of the matrix Rayle
   matrix&               S                   // computed vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::DynamicModeDecomposition(
   matrixf&              B,                  // second snapshot matrix in the pai
   ENUM_DMD_SCALE        jobs,               // determines whether the initial da
   ENUM_DMD_EIGV         jobz,               // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,               // determines whether to compute the
   ENUM_DMD_REFINE       jobf,               // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,             // allows for a selection of the SVD
   long                  nrnk,               // determines the mode how to comput
   float                 tol,                // the tolerance for truncating smal
   vectorcf&             eigen_values,       // vector of eigenvalues (Ritz value
   matrixf&              left_vectors,       // matrix of left singular vectors
   matrixf&              Z                   // matrix of Ritz vectors
   vectorf&              residuals,          // residuals for the Ritz pairs
   matrixf&              res_vectors,        // residual vectors for the Ritz pai
   matrixf&              B,                  // computed vectors
   matrixf&              W,                  // eigenvectors of the matrix Rayle
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixf&              S                       // computed vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::DynamicModeDecomposition(
   matrixc&              B,                      // second snapshot matrix in the pai
   ENUM_DMD_SCALE        jobs,                   // determines whether the initial da
   ENUM_DMD_EIGV         jobz,                   // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,                   // determines whether to compute the
   ENUM_DMD_REFINE       jobf,                   // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,                 // allows for a selection of the SVD
   long                  nrnk,                   // determines the mode how to comput
   double                tol,                    // the tolerance for truncating smal
   vectorc&              eigen_values,           // vector of eigenvalues (Ritz value
   matrixc&              left_vectors,           // matrix of left singular vectors
   matrixc&              Z                       // matrix of Ritz vectors
   vector&               residuals,              // residuals for the Ritz pairs
   matrixc&              res_vectors,            // residual vectors for the Ritz pai
   matrixc&              B,                      // computed vectors
   matrixc&              W,                      // eigenvectors of the matrix Rayle
   matrixc&              S                       // computed vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::DynamicModeDecomposition(
   matrixcf&             B,                      // second snapshot matrix in the pai
   ENUM_DMD_SCALE        jobs,                   // determines whether the initial da
   ENUM_DMD_EIGV         jobz,                   // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,                   // determines whether to compute the
   ENUM_DMD_REFINE       jobf,                   // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,                 // allows for a selection of the SVD
   long                  nrnk,                   // determines the mode how to comput
   float                 tol,                    // the tolerance for truncating smal
   vectorcf&             eigen_values,           // vector of eigenvalues (Ritz value
   matrixcf&             left_vectors,           // matrix of left singular vectors
   matrixcf&             Z                       // matrix of Ritz vectors
   vectorf&              residuals,              // residuals for the Ritz pairs
   matrixcf&             res_vectors,            // residual vectors for the Ritz pai
   matrixcf&             B,                      // computed vectors
   matrixcf&             W,                      // eigenvectors of the matrix Rayle
   matrixcf&             S                       // computed vectors
   );
```

Parameters

Y

[in] The second snapshot matrix in the pair.

© 2000-2025, MetaQuotes Ltd.

jobs

[in] Value from the ENUM_DMD_SCALE enumeration which determines whether the initial data snapshots are scaled by a diagonal matrix.

jobz

[in] Value from the ENUM_DMD_EIGV enumeration which determines whether the eigenvectors (Koopman modes) will be computed.

jobr

[in] Value from the ENUM_DMD_RESIDUALS enumeration which determines whether to compute the residuals.

jobf

[in] Value from the ENUM_DMD_REFINE enumeration which specifies whether to store information needed for post-processing (e.g. computing refined Ritz vectors).

whtsvd

[in] Value from the ENUM_SVD_ALG enumeration which allows for a selection of the SVD algorithm from the LAPACK library.

nrnk

[in] Value determines the mode how to compute the numerical rank, i.e. how to truncate small singular values of the input matrix X. If

NRNK = -1 :: i-th singular value sigma(i) is truncated if sigma(i) <= TOL*sigma(1). This option is recommended.

NRNK = -2 :: i-th singular value sigma(i) is truncated if sigma(i) <= TOL*sigma(i-1). This option is included for R&D purposes. It requires highly accurate SVD, which may not be feasible.

The numerical rank can be enforced by using positive value of NRNK as follows: 0 < NRNK <= N :: at most NRNK largest singular values will be used. If the number of the computed nonzero singular values is less than NRNK, then only those nonzero values will be used and the actually used dimension is less than NRNK.

tol

[in] The tolerance for truncating small singular values. 0 <= TOL < 1

eigen_values

[out] Vector of eigenvalues of size K. The leading K (K<=N) entries of EIGS contain the computed eigenvalues (Ritz values).

left_vectors

[out] Matrix of left singular vectors. The leading K columns contain a POD basis (POD - Proper Orthogonal Decomposition, правильная ортогональная декомпозиция), i.e. the leading K left singular vectors of the input data matrix X All N columns of contain all left singular vectors of the input matrix X.

Z

[out] Matrix of Ritz vectors. If JOBZ =='V' then Z contains the Ritz vectors. Z(:,i) is an eigenvector of the i-th Ritz value; ||Z(:,i)||_2=1. If JOBZ == 'F', then the Z(:,i)'s are given implicitly as the columns of X(:,1:K)*W(1:K,1:K), i.e. X(:,1:K)*W(:,i) is an eigenvector corresponding to EIGS(i). The columns of W(1:k,1:K) are the computed eigenvectors of the K-by-K Rayleigh quotient.

© 2000-2025, MetaQuotes Ltd.

residuals

[out] Residuals for the K computed Ritz pairs.

res_vectors

[out] Residual vectors for the K computed Ritz pairs.

B

[out] M-by-K matrix (K<=N).

If JOBF =='R', B(1:M,1:K) contains A*U(:,1:K), and can be used for computing the refined vectors; see further details in the provided references.

If JOBF == 'E', B(1:M,1:K) contains A*U(:,1:K)*W(1:K,1:K), which are the vectors from the Exact DMD, up to scaling by the inverse eigenvalues.

If JOBF =='N', then B is not referenced.

W

[out] K-by-K matrix (K<=N). Contains the K computed eigenvectors of the matrix Rayleigh quotient. The Ritz vectors (returned in Z) are the product of X (containing a POD basis for the input matrix X) and W.

S

[out] K-by-K matrix (K<=N). The array S(1:K,1:K) is used for the matrix Rayleigh quotient. This content is overwritten during the eigenvalue decomposition by GEEV.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The number of matrices rows (M) must not be less than the number of columns (N).

ENUM_DMD_SCALE

An enumeration that determines whether the initial data snapshots are scaled by a diagonal matrix.

ID	Description
DMDSCALE_S	'S': The data snapshots matrices X and Y are multiplied with a diagonal matrix D so that X*D has unit nonzero columns
DMDSCALE_C	'C': The snapshots are scaled as with the 'S' option.If it is found that an i-th column of X is zero vector and the corresponding i-th column of Y is non-zero, then the i-th column of Y is set to zero
DMDSCALE_Y	'Y': The data snapshots matrices X and Y are multiplied with a diagonal matrix D so that Y*D has unit nonzero columns
DMDSCALE_N	'N': No data scaling

© 2000-2025, MetaQuotes Ltd.

ENUM_DMD_EIGV

An enumeration which determines whether the eigenvectors (Koopman modes) will be computed.

ID	Description
DMDEIGV_V	'V': The eigenvectors (Koopman modes) will be computed
DMDEIGV_F	'F': The eigenvectors (Koopman modes) will be returned in factored form
DMDEIGV_N	'N': The eigenvectors are not computed

ENUM_DMD_RESIDUALS

An enumeration which determines whether to compute the residuals.

ID	Description
DMDRESIDUALS_R	'R': The residuals for the computed eigenpairs will be computed
DMDRESIDUALS_N	'N': The residuals are not computed

ENUM_DMD_REFINE

An enumeration which specifies whether to store information needed for post-processing.

ID	Description
DMDREFINE_R	'R': The matrix needed for the refinement of the Ritz vectors is computed and stored in the array B
DMDREFINE_E	'E': The unscaled eigenvectors of the Exact DMD are computed and returned in the array B
DMDREFINE_N	'N': No eigenvector refinement data is computed

ENUM_SVD_ALG

An enumeration selecting the SVD algorythm.

ID	Description
SVDALG_1	1: GESVD (the QR SVD algorithm)
SVDALG_2	2: GESDD (the Divide and Conquer algorithm)

© 2000-2025, MetaQuotes Ltd.

ID	Description
SVDALG_3	3: GESVDQ (the preconditioned QR SVD; this and 4 are the most accurate options)
SVDALG_4	4: GEJSV (the preconditioned Jacobi SVD; this and 3 are the most accurate options)

© 2000-2025, MetaQuotes Ltd.

DynamicModeDecompositionQR

Compute the Dynamic Mode Decomposition(по-русски "декомпозиция динамических мод") (DMD) for a pair of data snapshot matrices, using a QR factorization based compression of the data. Lapack function GEDMDQ. For the input matrices X and Y such that Y = A*X with an unaccessible matrix A, GEDMDQ computes a certain number of Ritz pairs of A using the standard Rayleigh-Ritz extraction from a subspace of range(X) that is determined using the leading left singular vectors of X. Optionally, GEDMDQ returns the residuals of the computed Ritz pairs, the information needed for a refinement of the Ritz vectors, or the eigenvectors of the Exact DMD.

The input M-by-N matrix F. The columns of F are the sequence of data snapshots from a single trajectory, taken at equidistant discrete times. It is assumed that the column norms of F are in the range of the normalized floating point numbers.

Computing for type matrix<double>

```
bool  matrix::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE        jobs,                 // determines whether the initial da
   ENUM_DMDQ_EIGV        jobz,                 // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,                 // determines whether to compute the
   ENUM_DMDQ_Q           jobq,                 // specifies whether to explicitly c
   ENUM_DMDQ_R           jobt,                 // specifies whether to return the u
   ENUM_DMD_REFINE       jobf,                 // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,               // allows for a selection of the SVD
   long                  nrnk,                 // determines the mode how to comput
   double                tol,                  // the tolerance for truncating smal
   vectorc&              eigen_values,         // vector of eigenvalues (Ritz value
   matrix&               Q,                    // Q factor from the QR factorizatio
   matrix&               R,                    // R factor from the QR factorizatio
   matrix&               left_vectors,         // matrix of left singular vectors
   matrix&               Z                     // matrix of Ritz vectors
   vector&               residuals,            // residuals for the Ritz pairs
   matrix&               B,                    // computed vectors
   matrix&               V,                    // eigenvectors of the matrix Rayle
   matrix&               S                     // computed vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE        jobs,                 // determines whether the initial da
   ENUM_DMDQ_EIGV        jobz,                 // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,                 // determines whether to compute the
   ENUM_DMDQ_Q           jobq,                 // specifies whether to explicitly c
   ENUM_DMDQ_R           jobt,                 // specifies whether to return the u
   ENUM_DMD_REFINE       jobf,                 // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,               // allows for a selection of the SVD
   long                  nrnk,                 // determines the mode how to comput
   float                 tol,                  // the tolerance for truncating smal
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorcf&            eigen_values,          // vector of eigenvalues (Ritz value
   matrixf&             Q,                     // Q factor from the QR factorizatio
   matrixf&             R,                     // R factor from the QR factorizatio
   matrixf&             left_vectors,          // matrix of left singular vectors
   matrixf&             Z                      // matrix of Ritz vectors
   vectorf&             residuals,             // residuals for the Ritz pairs
   matrixf&             B,                     // computed vectors
   matrixf&             V,                     // eigenvectors of the matrix Rayle
   matrixf&             S                      // computed vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE       jobs,                  // determines whether the initial da
   ENUM_DMDQ_EIGV       jobz,                  // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS   jobr,                  // determines whether to compute the
   ENUM_DMDQ_Q          jobq,                  // specifies whether to explicitly c
   ENUM_DMDQ_R          jobt,                  // specifies whether to return the u
   ENUM_DMD_REFINE      jobf,                  // specifies whether to store inform
   ENUM_SVD_ALG         whtsvd,                // allows for a selection of the SVD
   long                 nrnk,                  // determines the mode how to comput
   double               tol,                   // the tolerance for truncating smal
   vectorc&             eigen_values,          // vector of eigenvalues (Ritz value
   matrixc&             Q,                     // Q factor from the QR factorizatio
   matrixc&             R,                     // R factor from the QR factorizatio
   matrixc&             left_vectors,          // matrix of left singular vectors
   matrixc&             Z                      // matrix of Ritz vectors
   vector&              residuals,             // residuals for the Ritz pairs
   matrixc&             B,                     // computed vectors
   matrixc&             V,                     // eigenvectors of the matrix Rayle
   matrixc&             S                      // computed vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE       jobs,                  // determines whether the initial da
   ENUM_DMDQ_EIGV       jobz,                  // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS   jobr,                  // determines whether to compute the
   ENUM_DMDQ_Q          jobq,                  // specifies whether to explicitly c
   ENUM_DMDQ_R          jobt,                  // specifies whether to return the u
   ENUM_DMD_REFINE      jobf,                  // specifies whether to store inform
   ENUM_SVD_ALG         whtsvd,                // allows for a selection of the SVD
   long                 nrnk,                  // determines the mode how to comput
   float                tol,                   // the tolerance for truncating smal
   vectorcf&            eigen_values,          // vector of eigenvalues (Ritz value
   matrixcf&            Q,                     // Q factor from the QR factorizatio
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixcf&            R,                  // R factor from the QR factorizatio
   matrixcf&            left_vectors,       // matrix of left singular vectors
   matrixcf&            Z                   // matrix of Ritz vectors
   vectorf&             residuals,          // residuals for the Ritz pairs
   matrixcf&            B,                  // computed vectors
   matrixcf&            V,                  // eigenvectors of the matrix Rayle
   matrixcf&            S                   // computed vectors
   );
```

Parameters

jobs

[in] Value from the ENUM_DMD_SCALE enumeration which determines whether the initial data snapshots are scaled by a diagonal matrix. The data snapshots are the columns of F. The leading N-1 columns of F are denoted X and the trailing N-1 columns are denoted Y.

jobz

[in] Value from the ENUM_DMDQ_EIGV enumeration which determines whether the eigenvectors (Koopman modes) will be computed.

jobr

[in] Value from the ENUM_DMD_RESIDUALS enumeration which determines whether to compute the residuals.

jobq

[in] Value from the ENUM_DMDQ_Q enumeration which specifies whether to explicitly compute and return the unitary matrix from the QR factorization of the data snapshot matrix.

jobt

[in] Value from the ENUM_DMDQ_R enumeration which specifies whether to return the upper triangular factor from the QR factorization of the data snapshot matrix.

jobf

[in] Value from the ENUM_DMD_REFINE enumeration which specifies whether to store information needed for post-processing (e.g. computing refined Ritz vectors).

whtsvd

[in] Value from the ENUM_SVD_ALG enumeration which allows for a selection of the SVD algorithm from the LAPACK library.

nrnk

[in] Value determines the mode how to compute the numerical rank, i.e. how to truncate small singular values of the input matrix X. If

NRNK = -1 :: i-th singular value sigma(i) is truncated if sigma(i) <= TOL*sigma(1). This option is recommended.

NRNK = -2 :: i-th singular value sigma(i) is truncated if sigma(i) <= TOL*sigma(i-1). This option is included for R&D purposes. It requires highly accurate SVD, which may not be feasible.

The numerical rank can be enforced by using positive value of NRNK as follows: 0 < NRNK <= N-1 :: at most NRNK largest singular values will be used. If the number of the computed nonzero

© 2000-2025, MetaQuotes Ltd.

singular values is less than NRNK, then only those nonzero values will be used and the actually used dimension is less than NRNK.

tol

[in] The tolerance for truncating small singular values. 0 <= TOL < 1

eigen_values

[out] Vector of eigenvalues of size K. The leading K (K<N) entries of EIGS contain the computed eigenvalues (Ritz values).

Q

[out] Orthogonal matrix/factor of the QR factorization of the initial data snapshots matrix F.

R

[out] N-by-N upper triangular factor from the QR factorization of the data snapshot matrix F.

left_vectors

[out] Matrix of left singular vectors. The leading K columns of X contain the leading K left singular vectors of the hold representations of the leading N-1 snapshots in the orthonormal basis computed in the QR factorization of F. To lift them to the space of the left singular vectors U(:,1:K) of the input data, pre-multiply with the Q factor from the initial QR factorization.

Z

[out] M-by-K matrix

If JOBZ =='V' then Z contains the Ritz vectors. Z(:,i) is an eigenvector of the i-th Ritz value; ||Z(:,i)||_2=1.

If JOBZ == 'F', then the Z(:,i)'s are given implicitly as Z*V, where Z contains orthonormal matrix (the product of Q from the initial QR factorization and the SVD/POD_basis returned by GEDMD in X) and the second factor (the eigenvectors of the Rayleigh quotient) is in the matrix V, as returned by GEDMD. That is, X(:,1:K)*V(:,i) is an eigenvector corresponding to EIGS(i). The columns of V(1:K,1:K) are the computed eigenvectors of the K-by-K Rayleigh quotient.

residuals

[out] Residuals for the K computed Ritz pairs.

B

[out] N-by-K matrix (K<N).

If JOBF =='R', B(1:N,1:K) contains A*U(:,1:K), and can be used for computing the refined vectors.

If JOBF == 'E', B(1:N,1;K) contains A*U(:,1:K)*W(1:K,1:K), which are the vectors from the Exact DMD, up to scaling by the inverse eigenvalues.

In both cases, the content of B can be lifted to the original dimension of the input data by pre-multiplying with the Q factor from the initial QR factorization. Here A denotes a compression of the underlying operator.

If JOBF =='N', then B is not referenced.

V

[out] K-by-K matrix (K<N). Contains the the K eigenvectors of the Rayleigh quotient. The Ritz vectors (returned in Z) are the product of Q from the initial QR factorization.

© 2000-2025, MetaQuotes Ltd.

S

[out] K-by-K matrix (K<N). The array S(1:K,1:K) is used for the matrix Rayleigh quotient. This content is overwritten during the eigenvalue decomposition by GEEV.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The number of matrices rows (M) must not be less than the number of columns (N).

ENUM_DMD_SCALE

An enumeration that determines whether the initial data snapshots are scaled by a diagonal matrix.

ID	Description
DMDSCALE_S	'S': The data snapshots matrices X and Y are multiplied with a diagonal matrix D so that X*D has unit nonzero columns
DMDSCALE_C	'C': The snapshots are scaled as with the 'S' option.If it is found that an i-th column of X is zero vector and the corresponding i-th column of Y is non-zero, then the i-th column of Y is set to zero
DMDSCALE_Y	'Y': The data snapshots matrices X and Y are multiplied with a diagonal matrix D so that Y*D has unit nonzero columns
DMDSCALE_N	'N': No data scaling

ENUM_DMDQ_EIGV

An enumeration which determines whether the eigenvectors (Koopman modes) will be computed.

ID	Description
DMDQEIGV_V	'V': The eigenvectors (Koopman modes) will be computed
DMDQEIGV_F	'F': The eigenvectors (Koopman modes) will be returned in factored form as the product Z*V
DMDQEIGV_Q	'Q': The eigenvectors (Koopman modes) will be returned in factored form as the product Q*Z
DMDQEIGV_N	'N': The eigenvectors are not computed

ENUM_DMD_RESIDUALS

© 2000-2025, MetaQuotes Ltd.

An enumeration which determines whether to compute the residuals.

ID	Description
DMDRESIDUALS_R	'R': The residuals for the computed eigenpairs will be computed
DMDRESIDUALS_N	'N': The residuals are not computed

ENUM_DMDQ_Q

An enumeration which specifies whether to explicitly compute and return the unitary matrix from the QR factorization.

ID	Description
DMDQQ_Q	'Q': The matrix Q of the QR factorization of the data snapshot matrix is computed
DMDQQ_N	'N': The matrix Q is not explicitly computed

ENUM_DMDQ_R

An enumeration whichspecifies whether to return the upper triangular factor from the QR factorization.

ID	Description
DMDQR_R	'R': The matrix R of the QR factorization of the data snapshot matrix is computed
DMDQR_N	'N': The matrix R is not explicitly computed

ENUM_DMD_REFINE

An enumeration which specifies whether to store information needed for post-processing.

ID	Description
DMDREFINE_R	'R': The matrix needed for the refinement of the Ritz vectors is computed and stored in the array B
DMDREFINE_E	'E': The unscaled eigenvectors of the Exact DMD are computed and returned in the array B
DMDREFINE_N	'N': No eigenvector refinement data is computed

© 2000-2025, MetaQuotes Ltd.

ENUM_SVD_ALG

An enumeration selecting the SVD algorythm.

ID	Description
SVDALG_1	1: GESVD (the QR SVD algorithm)
SVDALG_2	2: GESDD (the Divide and Conquer algorithm)
SVDALG_3	3: GESVDQ (the preconditioned QR SVD; this and 4 are the most accurate options)
SVDALG_4	4: GEJSV (the preconditioned Jacobi SVD; this and 3 are the most accurate options)

© 2000-2025, MetaQuotes Ltd.

Singular Spectrum Analysis

The section contains functions for decomposing a matrix into three components: orthogonal matrices and a diagonal matrix of singular values. SVD is applied to solve various linear algebra problems such as data dimensionality reduction, image compression, solving systems of equations, as well as data analysis and optimization. The main functions allow calculating singular values and vectors, reconstruct matrices, as well as approximate them with reduced rank accuracy.

Function	Action
SingularSpectrumAnalysisSpectrum	A method function for calculating the relative contributions of spectral components based on their eigenvalues.
SingularSpectrumAnalysisForecast	A method function for calculating reconstructed and predicted data using spectral components of the input time series.
SingularSpectrumAnalysisReconstructComponents	A method function for calculating reconstructed components of the input time series and their contributions.
SingularSpectrumAnalysisReconstructSeries	A method function for calculating the reconstructed time series using the first component_count components.

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisSpectrum

A method function for calculating the relative contributions of spectral components based on their eigenvalues.

Calculations for vector<double> type

```
bool  vector::SingularSpectrumAnalysisSpectrum(
   ulong   window_length,      // window size for constructing the trajectory matrix
   vector& spectrum            // vector of component contributions to the input serie
   );
```

Calculations for vector<complex> type

```
bool  vector::SingularSpectrumAnalysisSpectrum(
   ulong   window_length,      // window size for constructing the trajectory matrix
   vectorc& spectrum           // vector of component contributions to the input serie
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

spectrum

[out] Vector of component contributions to the input series - eigenvalues of the covariance matrix of the input time series.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisForecast

A method function for calculating reconstructed and predicted data using spectral components of the input time series.

Calculations for vector<double> type

```
bool  vector::SingularSpectrumAnalysisForecast(
   ulong    window_length,         // window size for constructing the trajectory mat
   ulong    component_count,       // number of components used for forecasting
   ulong    forecast_horizon,      // number of points to forecast
   vector&  forecast               // vector consisting of reconstructed and predicte
   );
```

Calculations for vector<complex> type

```
bool  vector::SingularSpectrumAnalysisForecast(
   ulong    window_length,         // window size for constructing the trajectory mat
   ulong    component_count,       // number of components used for forecasting
   ulong    forecast_horizon,      // number of points to forecast
   vectorc& forecast               // vector consisting of reconstructed and predicte
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

component_count

[in] Number of components used for forecasting.

forecast_horizon

[in] Number of points to forecast.

forecast

[out] Combining points reconstructed by *component_count* plus *forecast_horizon* points predicted using the first *component_count* components. Thus, the *forecast* vector has the size of (T+forecast_horizon), where T is the input series length.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisReconstructComponents

A method function for calculating reconstructed components of the input time series and their contributions.

Calculations for vector<double> and matrix<double> types

```
bool  vector::SingularSpectrumAnalysisReconstructComponents(
   ulong    window_length,      // window size for constructing the trajectory matrix
   matrix&  components,         // matrix of reconstructed components
   vector&  contributions       // vector of component contributions to the input seri
   );
```

Calculations for vector<complex> and matrix<complex> types

```
bool  vector::SingularSpectrumAnalysisReconstructComponents(
   ulong    window_length,      // window size for constructing the trajectory matrix
   matrixc& components,         // matrix of reconstructed components
   vectorc& contributions       // vector of component contributions to the input seri
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

components

[out] A matrix of reconstructed components, where each column describes a separate component.

contributions

[out] Vector of component contributions to the input series (eigenvalues of the covariance matrix of the input time series).

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisReconstructSeries

A method function for calculating the reconstructed time series using the first *component_count* components.

Calculations for vector<double> type

```
bool  vector::SingularSpectrumAnalysisReconstructSeries(
   ulong    window_length,       // window size for constructing the trajectory matri
   ulong    component_count,     // number of components used for reconstruction
   vector&  reconstructed        // reconstructed time series
   );
```

Calculations for vector<complex> type

```
bool  vector::SingularSpectrumAnalysisReconstructSeries(
   ulong    window_length,       // window size for constructing the trajectory matri
   ulong    component_count,     // number of components used for reconstruction
   vectorc& reconstructed        // reconstructed time series
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

component_count

[out] Number of components used for time series reconstruction.

reconstructed

[out] Vector containing the reconstructed output series.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000-2025, MetaQuotes Ltd.

Reductions

Functions designed to transform matrices into special forms using orthogonal transformations. These methods are widely used in numerical computations, such as spectral analysis, eigenvalue problems, and singular value decomposition (SVD) of matrices.

These functions enable efficient matrix transformations essential for solving linear algebra problems, including computing Singular Value Decomposition (SVD) and finding eigenvalues of symmetric matrices. The use of LAPACK functions ensures high performance and computational accuracy.

Function	Action
ReduceToBidiagonal	Reduces a general real or complex m-by-n matrix A to upper or lower bidiagonal form B by an orthogonal transformation: Q**T * A * P = B. If m >= n, B is upper bidiagonal; if m < n, B is lower bidiagonal. Lapack function GEBRD.
ReflectBidiagonalToQP	Generates orthogonal matrices Q and P**T (or P**H for complex types) determined by ReduceToBidiagonal method when reducing a real or complex matrix A to bidiagonal form: A = Q * B * P**T. Q and P**T are defined as products of elementary reflectors H(i) or G(i) respectively. Lapack functions ORGBR, UNGBR.
ReduceSymmetricToTridiagonal	Reduces a real symmetric or complex Hermitian matrix A to trdiagonal form B by an orthogonal similarity transformation: Q**T * A * Q = B. Lapack functions SYTRD, HETRD.
ReflectTridiagonalToQ	Generates orthogonal matrix Q which is defined as the product of n-1 elementary reflectors of order n, as returned by ReduceSymmetricToTridiagonal

© 2000-2025, MetaQuotes Ltd.

ReduceToBidiagonal

Reduces a general real or complex m-by-n matrix A to upper or lower bidiagonal form B by an orthogonal transformation: Q**T * A * P = B. If m >= n, B is upper bidiagonal; if m < n, B is lower bidiagonal. Lapack function GEBRD.

Computing for type matrix<double>

```
bool  matrix::ReduceToBidiagonal(
   matrix&          B,             // bidiagonal matrix
   matrix&          reflect_qp,    // q and p-reflectors
   vector&          tau_q,         // scalar factors of the elementary reflectors Q
   vector&          tau_p          // scalar factors of the elementary reflectors P
   );
```

Computing for type matrix<float>

```
bool  matrix::ReduceToBidiagonal(
   matrixf&         B,             // bidiagonal matrix
   matrixf&         reflect_qp,    // q and p-reflectors
   vectorf&         tau_q,         // scalar factors of the elementary reflectors Q
   vectorf&         tau_p          // scalar factors of the elementary reflectors P
   );
```

Computing for type matrix<complex>

```
bool  matrix::ReduceToBidiagonal(
   matrixc&         B,             // bidiagonal matrix
   matrixc&         reflect_qp,    // q and p-reflectors
   vectorc&         tau_q,         // scalar factors of the elementary reflectors Q
   vectorc&         tau_p          // scalar factors of the elementary reflectors P
   );
```

Computing for type matrix<complexf>

```
bool  matrix::ReduceToBidiagonal(
   matrixcf&        B,             // bidiagonal matrix
   matrixcf&        reflect_qp,    // q and p-reflectors
   vectorcf&        tau_q,         // scalar factors of the elementary reflectors Q
   vectorcf&        tau_p          // scalar factors of the elementary reflectors P
   );
```

Parameters

B

[out] Upper or lower bidiagonal matrix.

reflect_qp

[out] Transformed matrix A. If m >= n, the diagonal and the first superdiagonal are overwritten with the upper bidiagonal matrix B; the elements below the diagonal, with the vector tau_q, represent the orthogonal matrix Q as a product of elementary reflectors, and the elements above the first superdiagonal, with the vector tau_p, represent the orthogonal matrix P as a product of

© 2000-2025, MetaQuotes Ltd.

elementary reflectors; if m < n, the diagonal and the first subdiagonal are overwritten with the lower bidiagonal matrix B; the elements below the first subdiagonal, with the vector tau_q, represent the orthogonal matrix Q as a product of elementary reflectors, and the elements above the diagonal, with the vector tau_p, represent the orthogonal matrix P as a product of elementary reflectors.

tau_q

[out] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

tau_p

[out] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix P.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The matrices Q and P are represented as products of elementary reflectors:

If m >= n,

Q = H(1) H(2) . . . H(n) and P = G(1) G(2) . . . G(n-1)

Each H(i) and G(i) has the form:

H(i) = I - tauq * v * v**T and G(i) = I - taup * u * u**T

where tauq and taup are scalars, and v and u are vectors;

v(1:i-1) = 0, v(i) = 1, and v(i+1:m) is stored on exit in A(i+1:m,i);

u(1:i) = 0, u(i+1) = 1, and u(i+2:n) is stored on exit in A(i,i+2:n);

tauq is stored in *tau_q*(i) and taup in *tau_p*(i).

If m < n,

Q = H(1) H(2) . . . H(m-1) and P = G(1) G(2) . . . G(m)

Each H(i) and G(i) has the form:

H(i) = I - tauq * v * v**T and G(i) = I - taup * u * u**T

where tauq and taup are scalars, and v and u are vectors;

v(1:i) = 0, v(i+1) = 1, and v(i+2:m) is stored on exit in A(i+2:m,i);

u(1:i-1) = 0, u(i) = 1, and u(i+1:n) is stored on exit in A(i,i+1:n);

tauq is stored in *tau_q*(i) and taup in *tau_p*(i).

The contents of A on exit (output matrix *reflect_qp*) are illustrated by the following examples:

© 2000-2025, MetaQuotes Ltd.

```
m = 6 and n = 5 (m > n):          m = 5 and n = 6 (m < n):
   (  d   e   u1  u1  u1 )           (  d   u1  u1  u1  u1  u1 )
   (  v1  d   e   u2  u2 )           (  e   d   u2  u2  u2  u2 )
   (  v1  v2  d   e   u3 )           (  v1  e   d   u3  u3  u3 )
   (  v1  v2  v3  d   e  )           (  v1  v2  e   d   u4  u4 )
   (  v1  v2  v3  v4  d  )           (  v1  v2  v3  e   d   u5 )
   (  v1  v2  v3  v4  v5 )
```

where d and e denote diagonal and off-diagonal elements of B, vi denotes an element of the vector defining H(i), and ui an element of the vector defining G(i).

Matrices Q and P can be produced with ReflectBidiagonalToQP method.

© 2000-2025, MetaQuotes Ltd.

ReflectBidiagonalToQP

Generates orthogonal matrices Q and P**T (or P**H for complex types) determined by ReduceToBidiagonal method when reducing a real or complex matrix A to bidiagonal form: A = Q * B * P**T. Q and P**T are defined as products of elementary reflectors H(i) or G(i) respectively. Lapack functions ORGBR, UNGBR.

As input is used transformed matrix reflect_qp with the same sizes m-by-n as in original matrix A.

Computing for type matrix<double>

```
bool  matrix::ReflectBidiagonalToQP(
   vector&          tau_q,         // scalar factors of the elementary reflectors Q
   vector&          tau_p,         // scalar factors of the elementary reflectors P
   matrix&          Q,             // matrix Q
   matrix&          PT             // transposed matrix P
   );
```

Computing for type matrix<float>

```
bool  matrix::ReflectBidiagonalToQP(
   vectorf&         tau_q,         // scalar factors of the elementary reflectors Q
   vectorf&         tau_p,         // scalar factors of the elementary reflectors P
   matrixf&         Q,             // matrix Q
   matrixf&         PT             // transposed matrix P
   );
```

Computing for type matrix<complex>

```
bool  matrix::ReflectBidiagonalToQP(
   vectorc&         tau_q,         // scalar factors of the elementary reflectors Q
   vectorc&         tau_p,         // scalar factors of the elementary reflectors P
   matrixc&         Q,             // matrix Q
   matrixc&         PH             // hermitian conjugated matrix P
   );
```

Computing for type matrix<complexf>

```
bool  matrix::ReflectBidiagonalToQP(
   vectorcf&        tau_q,         // scalar factors of the elementary reflectors Q
   vectorcf&        tau_p,         // scalar factors of the elementary reflectors P
   matrixcf&        Q,             // matrix Q
   matrixcf&        PH             // hermitian conjugated matrix P
   );
```

Parameters

tau_q

[in] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

tau_p

© 2000-2025, MetaQuotes Ltd.

[in] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix P.

Q

[out] Orthogonal matrix Q.

PT (PH)

[out] Transposed (or hermitian conjugated) matrix P.

Return Value

Return true if successful, otherwise false in case of an error.

Note

If m >= n, matrix Q is of m-by-n sizes, matrix PT is of n-by-n sizes.

If m < n, matrix Q is of m-by-m sizes, matrix PT is of m-by-n sizes.

© 2000-2025, MetaQuotes Ltd.

ReduceSymmetricToTridiagonal

Reduces a real symmetric or complex Hermitian matrix A to trdiagonal form B by an orthogonal similarity transformation: Q**T * A * Q = B. Lapack functions SYTRD, HETRD.

Computing for type matrix<double>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrix&          B,             // tridiagonal matrix
   matrix&          reflect_q,     // q-reflectors
   vector&          tau_q          // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<float>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrixf&         B,             // tridiagonal matrix
   matrixf&         reflect_q,     // q-reflectors
   vectorf&         tau_q          // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<complex>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrixc&         B,             // tridiagonal matrix
   matrixc&         reflect_q,     // q-reflectors
   vectorc&         tau_q          // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<complexf>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrixcf&        B,             // tridiagonal matrix
   matrixcf&        reflect_q,     // q and p-reflectors
   vectorcf&        tau_q          // scalar factors of the elementary reflectors Q
   );
```

Parameters

B

[out] Symmetric (or Hermitian) tridiagonal matrix.

reflect_q

[out] Upper or lower triangular matrix, it depends on the input matrix A. In upper case the diagonal and first superdiagonal of A are overwritten by the corresponding elements of the tridiagonal matrix B, and the elements above the first superdiagonal, with the array *tau_q*, represent the orthogonal matrix Q as a product of elementary reflectors; in lower case the diagonal and first subdiagonal of A are overwritten by the corresponding elements of the tridiagonal matrix B, and the elements below the first subdiagonal, with the array *tau_q*, represent the orthogonal matrix Q as a product of elementary reflectors.

tau_q

© 2000-2025, MetaQuotes Ltd.

[out] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

Note

In upper case the matrix Q is represented as a product of elementary reflectors

Q = H(n-1) . . . H(2) H(1)

Each H(i) has the form

H(i) = I - tau * v * v**T

where tau is a scalar, and v is a vector with v(i+1:n) = 0 and v(i) = 1; v(1:i-1) is stored on exit in A(1:i-1,i+1), and tau in *tau_q*(i).

In lower case the matrix Q is represented as a product of elementary reflectors

Q = H(1) H(2) . . . H(n-1)

Each H(i) has the form

H(i) = I - tau * v * v**T

where tau is a scalar, and v is a vector with v(1:i) = 0 and v(i+1) = 1; v(i+2:n) is stored on exit in A(i+2:n,i), and tau in *tau_q*(i). */

The contents of A on exit (output matrix *reflect_q*) are illustrated by the following examples with n = 5:

```
      upper case:                          lower case:

      (  d   e   v2  v3  v4 )              (  d                   )
      (      d   e   v3  v4 )              (  e   d               )
      (          d   e   v4 )              (  v1  e   d           )
      (              d   e  )              (  v1  v2  e   d       )
      (                  d  )              (  v1  v2  v3  e   d   )
```

where d and e denote diagonal and off-diagonal elements of B and vi denotes an element of the vector defining H(i).

Matrix Q can be produced with ReflectTridiagonalToQ method.

© 2000-2025, MetaQuotes Ltd.

ReflectTridiagonalToQ

Generates orthogonal matrix Q which is defined as the product of n-1 elementary reflectors of order n, as returned by ReduceSymmetricToTridiagonal:

in upper case Q = H(n-1) . . . H(2) H(1),

in lower case Q = H(1) H(2) . . . H(n-1).

Lapack functions ORGTR, UNGTR.

As input is used transformed matrix reflect_q with the same sizes n-by-n as in original matrix A.

Computing for type matrix<double>

```
bool  matrix::ReflectTridiagonalToQ(
   vector&          tau_q,         // scalar factors of the elementary reflectors Q
   matrix&          Q              // matrix Q
   );
```

Computing for type matrix<float>

```
bool  matrix::ReflectTridiagonalToQ(
   vectorf&         tau_q,         // scalar factors of the elementary reflectors Q
   matrixf&         Q              // matrix Q
   );
```

Computing for type matrix<complex>

```
bool  matrix::ReflectTridiagonalToQ(
   vectorc&         tau_q,         // scalar factors of the elementary reflectors Q
   matrixc&         Q              // matrix Q
   );
```

Computing for type matrix<complexf>

```
bool  matrix::ReflectTridiagonalToQ(
   vectorcf&        tau_q,         // scalar factors of the elementary reflectors Q
   matrixcf&        Q              // matrix Q
   );
```

Parameters

tau_q

[in] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

Q

[out] Orthogonal matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Conversión de datos

Se trata de un grupo de funciones encargadas de transformar datos de un formato al otro.

Sobre todo hay que destacar la función NormalizeDouble() que proporciona la precisión necesaria a la hora representar el precio. En las operaciones comerciales no se puede utilizar los precios no normalizados cuya precisión supera la requerida por el servidor comercial, aunque sea por un dígito.

Función	Acción
CharToString	Conversión del código del símbolo a una cadena de un carácter
DoubleToString	Conversión del valor numérico a una cadena de caracteres con una precisión especificada
EnumToString	Conversión del valor de una enumeración de cualquier tipo a una cadena de caracteres
NormalizeDouble	Redondeo de un número con punto flotante hasta una precisión especificada
StringToDouble	Conversión de una cadena que contiene la representación simbólica de un número a un número del tipo double
StringToInteger	Conversión de una cadena que contiene la representación simbólica de un número a un número del tipo long
StringToTime	Conversión de una cadena que contiene la hora y/o la fecha en el formato "yyyy.mm.dd [hh:mi]" al número del tipo datetime
TimeToString	Conversión del valor que contiene el tiempo en segundos transcurridos desde el 01.01.1970 a la cadena con el formato "yyyy.mm.dd hh:mi"
IntegerToString	Conversión del valor del tipo entero a una cadena de longitud definida
ShortToString	Conversión del código del símbolo (unicode) a una cadena de un carácter
ShortArrayToString	Copia una parte del array en una cadena
StringToShortArray	Copia caracteres de una cadena en una parte especificada de un array del tipo ushort
CharArrayToString	Conversión del código del símbolo (ansi) a una cadena de un carácter
StringToCharArray	Copia los caracteres de una cadena transformada de Unicode en ANSI en una parte seleccionada de un array del tipo uchar
CharArrayToStruct	Copia una matriz del tipo uchar a una estructura POD
StructToCharArray	Copia una estructura POD a una matriz de tipo uchar
ColorToARGB	Conversión del tipo color al tipo uint para conseguir la representación del color ARGB.

© 2000-2025, MetaQuotes Ltd.

Función	Acción
ColorToString	Conversión de valores de colores a una cadena del tipo "R,G,B"
StringToColor	Conversión de una cadena del tipo "R,G,B" o una cadena que contiene nombre de un color al valor del tipo color
StringFormat	Conversión de un número a una cadena conforme al formato especificado

Véase también

Uso de página de código

© 2000-2025, MetaQuotes Ltd.

CharToString

Conversión del código del símbolo a una cadena de un carácter.

```
string  CharToString(
   uchar  char_code      // código numérico del símbolo
   );
```

Parámetros

char_code

[in] Código del símbolo ANSI.

Valor devuelto

Cadena que contiene un símbolo ANSI.

Ejemplo:

```
string ExtStrArray[224];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- en un ciclo desde el código de caracter 32(espacio) hasta el código 255(UCHAR_MA
//--- rellenamos el array con códigos de caracteres convertidos a cadena según la pág
   for(int i=32; i<=UCHAR_MAX; i++)
      ExtStrArray[i-32]=CharToString((uchar)i);

//--- imprimimos en el registro de encabezados y el recuadro de caracteres
   Print("Table of symbols:");
   ArrayPrint(ExtStrArray,_Digits," | ");
   /*
  resultado:
   Table of symbols:
   [  0] " " | "!" | """ | "#" | "$" | "%" | "&" | "'" | "(" | ")" | "*" | "+" | ","
   [ 25] "9" | ":" | ";" | "<" | "=" | ">" | "?" | "@" | "A" | "B" | "C" | "D" | "E"
   [ 50] "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "[" | "\" | "]" | "^"
   [ 75] "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w"
   [100] "„" | "…" | "†" | "‡" | "€" | "‰" | "Љ" | "‹" | "Њ" | "Ќ" | "Ћ" | "Џ" | "ђ"
   [125] "ќ" | "ћ" | "џ" | " " | "ў" | "Ў" | "Ј" | "¤" | "Ґ" | "¦" | "§" | "Ё" | "©"
   [150] "¶" | "·" | "ё" | "№" | "є" | "»" | "ј" | "S" | "s" | "ї" | "A" | "Б" | "В"
   [175] "П" | "Р" | "С" | "Т" | "У" | "Ф" | "Х" | "Ц" | "Ч" | "Ш" | "Щ" | "Ъ" | "Ы"
   [200] "и" | "й" | "к" | "л" | "м" | "н" | "о" | "п" | "р" | "с" | "т" | "у" | "ф"
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Véase también

StringToCharArray, ShortToString, StringGetCharacter

© 2000-2025, MetaQuotes Ltd.

CharArrayToString

Copia y transforma una parte del array del tipo uchar en una cadena devuelta.

```
string  CharArrayToString(
   uchar    array[],             // array
   int      start=0,             // posición inicial en el array
   int      count=-1             // número de símbolos
   uint     codepage=CP_ACP      // página de código
   );
```

Parámetros

array[]

[in] Array del tipo uchar.

start=0

[in] Posición de que se empieza a copiar. Por defecto es 0.

count=-1

[in] Número de elementos del array a copiar. Determina la longitud de la cadena de resultado. Por defecto es -1, lo que significa el copiado hasta el final del array, o hasta encontrarse con el 0 de terminación.

codepage=CP_ACP

[in] Valor de la página de código. Para las páginas de código más usadas están previstas unas constantes correspondientes.

Valor devuelto

Una cadena.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- creamos un array con el código de los caracteres
   uchar array[]= { 84, 104, 105, 115,  32, 105, 115,  32,  97,  32,
                    116, 101, 115, 116,  32, 111, 102,  32, 116, 104,
                    101,  32,  67, 104,  97, 114,  65, 114, 114,  97,
                    121,  84, 111,  83, 116, 114, 105, 110, 103,  40,
                     41,  32, 102, 117, 110,  99, 116, 105, 111, 110
                   };
//--- imprimimos en el registro un array de códigos transformados en una cadena
   Print(CharArrayToString(array));

//--- creamos un array con el código de los caracteres y terminal cero
```

© 2000-2025, MetaQuotes Ltd.

```
   uchar array_z[]= { 84, 104, 105, 115,  32, 105, 115,  32,  97,  32,
                     116, 101, 115, 116,   0,  32, 111, 102,  32, 116,
                     104, 101,  32,  67, 104,  97, 114,  65, 114, 114,
                      97, 121,  84, 111,  83, 116, 114, 105, 110, 103,
                      40,  41,  32, 102, 117, 110,  99, 116, 105, 111, 110
                    };
//--- imprimimos en el registro el segundo array de códigos con terminal cero
   PrintFormat("%s ...", CharArrayToString(array_z));

   /*
  resultado:
   This is a test of the CharArrayToString() function
   This is a test ...
   */
  }
```

Véase también

StringToCharArray, ShortArrayToString, Uso de página de código

© 2000-2025, MetaQuotes Ltd.

CharArrayToStruct

Copia una matriz del tipo uchar a una estructura POD.

```
bool  CharArrayToStruct(
   void&          struct_object,     // estructura
   const uchar&   char_array[],      // matriz
   uint           start_pos=0        // posición inicial en la matriz
   );
```

Parámetros

struct_object

[in] Enlace a cualquier tipo de estructura POD (estructura que contiene solo tipos simples de datos).

char_array[]

[in] Matriz del tipo uchar.

start_pos=0

[in] Posición en la matriz, a partir de la cual se copiarán los datos.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Ejemplo:

```
#define DATA_TOTAL 4

MqlRates ExtDataRates[DATA_TOTAL];
uchar    ExtCharArray[];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- copiamos en un array uchar los datos de las cuatro últimas barras
   ResetLastError();
   for(int i=0; i<DATA_TOTAL; i++)
     {
      if(!CopyRatesToCharArray(i, ExtCharArray))
         return;
     }

//--- copiamos todos los datos disponibles de un array uchar a un array de estructuras
   for(int i=0; i<DATA_TOTAL; i++)
     {
      ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
      if(!CharArrayToStruct(ExtDataRates[i], ExtCharArray, sizeof(MqlRates)*i))
        {
         Print("CharArrayToStruct() failed. Error code: ", GetLastError());
         return;
        }
      //--- si los datos se copian correctamente del array uchar a la estructura MqlRa
      //--- imprimimos en el registro los datos copiados en la estructura
      MqlRatesPrint(ExtDataRates[i]);
     }

  /*
resultado:
 MqlRates data:
  Time = 2024.02.21 09:00;
  Open = 1.08116;
  High = 1.08141;
  Low = 1.08062;
  Close = 1.08124;
  Tick volume = 5344;
  Spread = 1;
  Real volume = 0

 MqlRates data:
  Time = 2024.02.21 08:00;
  Open = 1.08149;
  High = 1.08153;
  Low = 1.08073;
  Close = 1.08114;
  Tick volume = 3607;
  Spread = 2;
  Real volume = 0

 MqlRates data:
  Time = 2024.02.21 07:00;
  Open = 1.08143;
  High = 1.08165;
  Low = 1.08122;
  Close = 1.08150;
  Tick volume = 2083;
  Spread = 0;
  Real volume = 0

 MqlRates data:
  Time = 2024.02.21 06:00;
  Open = 1.08178;
  High = 1.08190;
  Low = 1.08132;
  Close = 1.08142;
  Tick volume = 1733;
```

© 2000-2025, MetaQuotes Ltd.

```
    Spread = 0;
    Real volume = 0
   */
  }
//+------------------------------------------------------------------+
//| Según el índice indicado, copia los datos de la barra            |
//| a través de la estructura MqlRates en un array uchar             |
//+------------------------------------------------------------------+
bool CopyRatesToCharArray(const int index, uchar &data_array[])
  {
   MqlRates rates[1];
   uint     start=sizeof(MqlRates);

   ResetLastError();
   if(CopyRates(Symbol(), PERIOD_CURRENT, index, 1, rates)!=1)
     {
      Print("CopyRates() failed. Error code: ", GetLastError());
      return(false);
     }
   if(!StructToCharArray(rates[0], data_array, start*index))
     {
      Print("StructToCharArray() failed. Error code: ", GetLastError());
      return(false);
     }

   return(true);
  }
//+------------------------------------------------------------------+
//| Imprime en el registro los campos de la estructura MqlRates      |
//+------------------------------------------------------------------+
void MqlRatesPrint(MqlRates &rates)
  {
//--- creamos una línea de muestra
   string text=StringFormat(" Time = %s;\n"
                            " Open = %.*f;\n"
                            " High = %.*f;\n"
                            " Low = %.*f;\n"
                            " Close = %.*f;\n"
                            " Tick volume = %I64u;\n"
                            " Spread = %d;\n"
                            " Real volume = %I64u",
                            TimeToString(rates.time),
                            _Digits, rates.open,
                            _Digits, rates.high,
                            _Digits, rates.low,
                            _Digits, rates.close,
                            rates.tick_volume,
                            rates.spread,
                            rates.real_volume);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- imprimimos en el registro de encabezados y la línea de muestra
   Print("MqlRates data:\n", text, "\n");
  }
```

Ver también

StringToCharArray, ShortArrayToString, StructToCharArray, Uso de la página de código, FileReadStruct, Unión (union), MathSwap

© 2000-2025, MetaQuotes Ltd.

StructToCharArray

Copia una estructura POD a una matriz de tipo uchar.

```
bool  StructToCharArray(
   const void&  struct_object,     // estructura
   uchar&       char_array[],      // matriz
   uint         start_pos=0        // posición inicial en la matriz
   );
```

Parámetros

struct_object

[in] Enlace a cualquier tipo de estructura POD (estructura que contiene solo tipos simples de datos).

char_array[]

[in] Matriz del tipo uchar.

start_pos=0

[in] Posición en la matriz, a partir de la cual se copiarán los datos.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false.

Observación

Durante el copiado, la matriz dinámica se expande automáticamente (ArrayResize), si en ella no hay espacio suficiente. Si no se ha logrado expandir la matriz hasta el tamaño necesario, la función retornará error.

Ejemplo:

```
uchar    ExtCharArray[];
MqlRates ExtRates[1];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- copiamos los datos de la barra actual en la estructura MqlRates
   if(CopyRates(Symbol(), PERIOD_CURRENT, 0, 1, ExtRates)!=1)
     {
      Print("CopyRates() failed. Error code: ", GetLastError());
      return;
     }

//--- imprimimos en el registro los datos recibidos en la estructura MqlRates
   Print("Data in the structure immediately after receiving it:");
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlRatesPrint(ExtRates[0]);

//--- copiamos la estructura en una array del tipo uchar
   ResetLastError();
   if(!StructToCharArray(ExtRates[0], ExtCharArray))
     {
      Print("StructToCharArray() failed. Error code: ", GetLastError());
      return;
     }

//--- limpiamos la estructura
   ZeroMemory(ExtRates[0]);
//--- imprimimos los datos en la estructura después de realizar la limpieza
   Print("\nData in the structure after ZeroMemory():");
   MqlRatesPrint(ExtRates[0]);

//--- ahora compiamos los datos del array uchar a la estructura MqlRates
   if(!CharArrayToStruct(ExtRates[0], ExtCharArray))
     {
      Print("CharArrayToStruct() failed. Error code: ", GetLastError());
      return;
     }
//--- imprimimos los datos en la estructura MqlRates después de copiar del array uchar
   Print("\nData in the MqlRates structure after copying it from a uchar array:");
   MqlRatesPrint(ExtRates[0]);

   /*
  resultado:

   Data in the structure immediately after receiving it:
   MqlRates data:
    Time = 2024.02.21 07:00;
    Open = 1.08143;
    High = 1.08158;
    Low = 1.08122;
    Close = 1.08137;
    Tick volume = 1341;
    Spread = 4;
    Real volume = 0

   Data in the structure after ZeroMemory():
   MqlRates data:
    Time = 0;
    Open = 0.00000;
    High = 0.00000;
    Low = 0.00000;
    Close = 0.00000;
    Tick volume = 0;
    Spread = 0;
```

© 2000-2025, MetaQuotes Ltd.

```
    Real volume = 0

   Data in the MqlRates structure after copying it from a uchar array:
   MqlRates data:
    Time = 2024.02.21 07:00;
    Open = 1.08143;
    High = 1.08158;
    Low = 1.08122;
    Close = 1.08137;
    Tick volume = 1341;
    Spread = 4;
    Real volume = 0
   */
  }
//+------------------------------------------------------------------+
//| Imprime en el registro los campos de la estructura MqlRates      |
//+------------------------------------------------------------------+
void MqlRatesPrint(MqlRates &rates)
  {
//--- creamos una línea de muestra
   string text=StringFormat(" Time = %s;\n"
                            " Open = %.*f;\n"
                            " High = %.*f;\n"
                            " Low = %.*f;\n"
                            " Close = %.*f;\n"
                            " Tick volume = %I64u;\n"
                            " Spread = %d;\n"
                            " Real volume = %I64u",
                            TimeToString(rates.time),
                            _Digits, rates.open,
                            _Digits, rates.high,
                            _Digits, rates.low,
                            _Digits, rates.close,
                            rates.tick_volume,
                            rates.spread,
                            rates.real_volume);
//--- imprimimos en el registro de encabezados y la línea de muestra
   Print("MqlRates data:\n", text);
  }
```

Ver también

StringToCharArray, ShortArrayToString, CharArrayToStruct, Uso de la página de código, FileWriteStruct, Unión (union), MathSwap

© 2000-2025, MetaQuotes Ltd.

ColorToARGB

Esta función convierte el tipo color al tipo uint para conseguir la representación de color ARGB. El formato de color ARGB se utiliza durante la creación de recursos gráficos, visualización del texto y en la clase de la biblioteca estándar CCanvas.

```
uint  ColorToARGB(
   color  clr,          // color en el formato color para convertir
   uchar  alpha=255     // composición alfa que responde del grado de transparencia de
   );
```

Parámetros

clr

[in] Valor del color en la variable del tipo color.

alpha

[in] Valor de composición alfa para obtener el color en el formato ARGB. Se puede establecer el valor de 0 (el color del píxel puesto encima no cambia en absoluto la visualización del píxel del fondo) hasta 255 (el color se pone encima íntegramente y cubre totalmente el color del píxel del fondo). La transparencia del color en términos porcentuales se calcula según la siguiente fórmula (1-alpha/255)*100%. Es decir, cuanto menor sea el valor de la composición alfa, más transparente será el color.

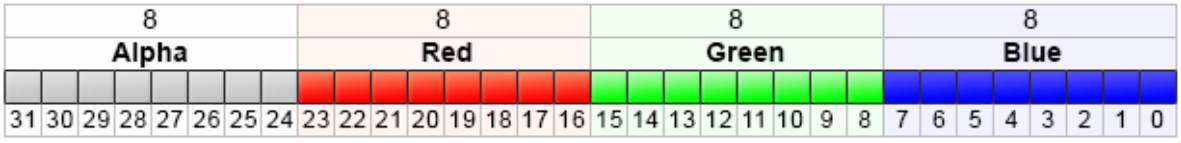
Valor devuelto

La representación del color en el formato ARGB, donde los valores Alfa, Red, Green, Blue (composición alfa, rojo, verde, azul) están escritos por orden en cuatro bytes del tipo uint.

Nota

RGB es el formato básico y mayoritariamente utilizado que sirve para describir el color del píxel sobre la pantalla en la computación gráfica. Los nombres de colores básicos se utilizan para establecer los componentes del color: rojo (Red), verde(Green) y azul (Blue). Cada componente se describe con un byte que establece la saturación de este color en un intervalo de 0 a 255 (de 0x00 a 0XFF en el formato hexadecimal). Puesto que el color blanco contiene todos los colores, se describe como 0xFFFFFF. O sea, cada uno de tres componentes está representado aquí con el valor máximo 0xFF.

Sin embargo, en algunas tareas se requiere la especificación de la transparencia del color con el fin de describir qué apariencia va a tener la imagen si la cubren con un color con cierto grado de transparencia. Para estas ocasiones se introduce el concepto de la composición alfa como un componente adicional al formato RGB. El esquema del formato ARGB se muestra en la figura de abajo.



Los valores ARGB habitualmente se indican en el formato hexadecimal, donde cada par de cifras representa por orden los valores de las composiciones Alpha, Red, Green y Blue. Por ejemplo, el color-ARGB 80FFFF00 representa el amarillo con opacidad de 50,2 %. Al principio va 0x80 que

© 2000-2025, MetaQuotes Ltd.

establece el 50,2% de la composición alfa, ya que es un 50,2% del valor 0xFF. Luego el primer par FF representa el valor máximo del componente rojo; el siguiente par FF establece la misma intensidad del componente verde; y el último par 00 representa el mínimo valor del componente azul (ausencia del azul). La combinación del verde y el rojo produce el amarillo. Si la composición alfa no se utiliza, la entrada puede ser reducida a 6 dígitos RRGGBB. Por esa razón los valores de la composición alfa se guardan en los bits superiores del tipo de números enteros uint.

En función del contexto, los números hexadecimales pueden escribirse con el prefijo '0x' o '#', por ejemplo, 80FFFF00, o 0x80FFFF00, o #80FFFF00.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- establecemos la transparencia
   uchar alpha=0x55;  // el valor 0x55 significa 55/255=21,6 % de transparencia
   //--- obtenemos la conversión a ARGB para el color clrBlue
   PrintFormat("0x%.8X - clrBlue",clrBlue);
   PrintFormat("0x%.8X - clrBlue ARGB with alpha=0x55 (transparency 21.6%%)",ColorToAR
   //--- obtenemos la conversión a ARGB para el color clrGreen
   PrintFormat("0x%.8X - clrGreen",clrGreen);
   PrintFormat("0x%.8X - clrGreen ARGB with alpha=0x55 (transparency 21.6%%)",ColorToA
   //--- obtenemos la conversión a ARGB para el color clrRed
   PrintFormat("0x%.8X - clrRed",clrRed);
   PrintFormat("0x%.8X - clrRed ARGB with alpha=0x55 (transparency 21.6%%)",ColorToARG
  }
```

Véase también

Recursos, ResourceCreate(), TextOut(), Tipo color, Tipos char, short, int y long

© 2000-2025, MetaQuotes Ltd.

ColorToString

Conversión de un valor de colores a una cadena del tipo "R,G,B".

```
string  ColorToString(
   color  color_value,     // valor de color
   bool   color_name       // mostrar nombre de color o no
   );
```

Parámetros

color_value

[in] Valor del color en la variable del tipo color.

color_name

[in] La señal de necesidad de devolver el nombre del color, si el valor del color coincide con una de las constantes de color predefinidas.

Valor devuelto

Representación literal de color como "R,G,B", donde R, G y B son constantes decimales de 0 a 255 convertidas a una cadena. Si el parámetro color_name=true está definido, se intenta convertir el valor del color al nombre del color.

Ejemplo:

```
   string clr=ColorToString(C'0,255,0'); // color verde
   Print(clr);

   clr=ColorToString(C'0,255,0',true);   // recibir constante de color
   Print(clr);
```

Véase también

StringToColor, ColorToARGB

© 2000-2025, MetaQuotes Ltd.

DoubleToString

Conversión del valor numérico a una cadena de caracteres.

```
string  DoubleToString(
   double  value,      // número
   int     digits=8    // número de dígitos después del punto decimal
   );
```

Parámetros

value

[in] Valor con punto flotante.

digits

[in] Formato de precisión. Si el valor digits se encuentra dentro del rango de 0 a 16, obtenemos la representación literal del número con la cantidad especificada de dígitos después del punto. Si el valor digits se encuentra dentro del rango de -1 a -16, obtenemos la representación literal del número en el formato científico con la cantidad especificada de dígitos después del punto. En todos los demás casos, el valor literal del número tendrá 8 dígitos después del punto.

Valor devuelto

Cadena que contiene representación de símbolos del número en el formato de precisión especificado.

Ejemplo:

```
   Print("DoubleToString(120.0 + M_PI) : ",DoubleToString(120.0+M_PI));
   Print("DoubleToString(120.0 + M_PI,16) : ",DoubleToString(120.0+M_PI,16));
   Print("DoubleToString(120.0 + M_PI,-16) : ",DoubleToString(120.0+M_PI,-16));
   Print("DoubleToString(120.0 + M_PI,-1) : ",DoubleToString(120.0+M_PI,-1));
   Print("DoubleToString(120.0 + M_PI,-20) : ",DoubleToString(120.0+M_PI,-20));
```

Véase también

NormalizeDouble, StringToDouble

© 2000-2025, MetaQuotes Ltd.

EnumToString

Convierte el valor de una enumeración de cualquier tipo a una cadena de caracteres.

```
string  EnumToString(
   any_enum  value      // valor de la enumeración de cualquier tipo
   );
```

Parámetros

value

[in] Valor de la enumeración de cualquier tipo.

Valor devuelto

Una cadena que contiene la representación de texto del valor. Para obtener la información sobre el error, hay que llamara a la función GetLastError().

Nota

La función GetLastError() puede establecer los siguientes valores en la variable _LastError:

- ERR_INTERNAL_ERROR - error del entorno de ejecución
- ERR_NOT_ENOUGH_MEMORY - no hay suficiente memoria para completar la operación
- ERR_INVALID_PARAMETER - no se puede aceptar el nombre del valor de la enumeración

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
enum interval  // enumeración de constantes nombradas
  {
   month=1,     // período de un mes
   two_months,  // dos meses
   quarter,     // tres meses - trimestre
   halfyear=6,  // semestre
   year=12,     // año - 12 meses
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- se establece el período de tiempo que es igual a un mes
   interval period=month;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- se establece el período de tiempo que es igual a un trimestre (tres meses))
   period=quarter;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- se establece el período de tiempo que es igual a un año (12 meses))
   period=year;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- comprobamos cómo se muestra el tipo de la orden
   ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
   Print(EnumToString(type)+"="+IntegerToString(type));

//--- comprobamos cómo se muestra el valor incorrecto
   type=WRONG_VALUE;
   Print(EnumToString(type)+"="+IntegerToString(type));

// Resultado de ejecución:
// month=1
// quarter=3
// year=12
// ORDER_TYPE_BUY=0
// ENUM_ORDER_TYPE::-1=-1
  }
```

Véase también

Enumeraciones, Variables de entrada Input

© 2000-2025, MetaQuotes Ltd.

IntegerToString

Convierte el valor del tipo entero a una cadena de longitud especificada y devuelve la cadena obtenida.

```
string  IntegerToString(
   long     number,              // número
   int      str_len=0,           // longitud de la cadena en la salida
   ushort   fill_symbol=' '      // relleno
   );
```

Parámetros

number

[in] Número para conversión.

str_len=0

[in] Longitud de la cadena. Si la longitud de la cadena obtenida resulta ser más de la especificada, la cadena no se recorta. Si la longitud de la cadena obtenida resulta ser menos de la especificada, los símbolos de relleno se añadirán a esta cadena por la izquierda.

fill_symbol=' '

[in] Símbolo de relleno. Por defecto es un espacio.

Valor devuelto

Cadena.

Ejemplo:

```
#define DATA_TOTAL 1001

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- en un ciclo por DATA_TOTAL, mostramos las líneas con un índice divisible por 100
//--- en la línea se muestra el valor del índice en formato de cuatro caracteres
//--- si la longitud de la cadena es inferior a 4 caracteres, el valor del índice del
//--- en la línea va precedido de ceros a la izquierda
   for(int i=0; i<DATA_TOTAL; i++)
     {
      if(i%100==0)
         Print("Converted index value: ",IntegerToString(i,4,'0'));
     }
   /*
  resultado:
   Converted index value: 0000
   Converted index value: 0100
   Converted index value: 0200
```

© 2000-2025, MetaQuotes Ltd.

```
   Converted index value: 0300
   Converted index value: 0400
   Converted index value: 0500
   Converted index value: 0600
   Converted index value: 0700
   Converted index value: 0800
   Converted index value: 0900
   Converted index value: 1000
   */
  }
```

Véase también

StringToInteger

© 2000-2025, MetaQuotes Ltd.

ShortToString

Convierte el código del símbolo (unicode) a una cadena de un carácter y devuelve la cadena obtenida.

```
string  ShortToString(
   ushort  symbol_code      // símbolo
   );
```

Parámetros

symbol_code

[in] Código del símbolo. En vez del código del símbolo se puede usar una cadena literal que contiene un símbolo, o una cadena literal con un código hexadecimal de dos bytes que corresponde al código de la tabla Unicode.

Valor devuelto

Cadena.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- en un ciclo, imprimiremos 18 caracteres, comenzando con el carácter con el núme
   for(int i=0; i<18; i++)
     {
      ushort code=0x23E9+(ushort)i;
      PrintFormat("Unicode number U+%hX: %s",code,ShortToString(code));
     }
   /*
  resultado:
   Unicode number U+23E9: ⏩
   Unicode number U+23EA: ⏪
   Unicode number U+23EB: ⏫
   Unicode number U+23EC: ⏬
   Unicode number U+23ED: ⏭
   Unicode number U+23EE: ⏮
   Unicode number U+23EF: ⏯
   Unicode number U+23F0: ⏰
   Unicode number U+23F1: ⏱
   Unicode number U+23F2: ⏲
   Unicode number U+23F3: ⏳
   Unicode number U+23F4: □
   Unicode number U+23F5: □
   Unicode number U+23F6: □
   Unicode number U+23F7: □
```

© 2000-2025, MetaQuotes Ltd.

```
   Unicode number U+23F8: ⏸
   Unicode number U+23F9: ⏹
   Unicode number U+23FA: ⏺
   */
  }
```

Véase también

StringToCharArray, CharToString, StringGetCharacter

© 2000-2025, MetaQuotes Ltd.

ShortArrayToString

Copia una parte del array en la cadena devuelta.

```
string  ShortArrayToString(
   ushort  array[],      // array
   int     start=0,      // posición inicial en el array
   int     count=-1      // cantidad de símbolos
   );
```

Parámetros

array[]

[in] Array del tipo ushort (análogo del tipo wchar_t).

start=0

[in] Punto de partida del copiado. Por defecto es 0.

count=-1

[in] Número de elementos del array para ser copiados. Determina la longitud de la cadena resultante. El valor por defecto es -1, esto significa el copiado hasta el final del array, o hasta el 0 de terminación.

Valor devuelto

Cadena.

Ejemplo:

```
#define ROW_SIZE 16

ushort ExtShortArray[]={0x2190,0x2191,0x2192,0x2193,0x2194,0x2195,0x2196,0x2197,0x2198
                        0x21A0,0x21A1,0x21A2,0x21A3,0x21A4,0x21A5,0x21A6,0x21A7,0x21A8
                        0x21B0,0x21B1,0x21B2,0x21B3,0x21B4,0x21B5,0x21B6,0x21B7,0x21B8
                        0x21C0,0x21C1,0x21C2,0x21C3,0x21C4,0x21C5,0x21C6,0x21C7,0x21C8
                        0x21D0,0x21D1,0x21D2,0x21D3,0x21D4,0x21D5,0x21D6,0x21D7,0x21D8
                        0x21E0,0x21E1,0x21E2,0x21E3,0x21E4,0x21E5,0x21E6,0x21E7,0x21E8
                        0x21F0,0x21F1,0x21F2,0x21F3,0x21F4,0x21F5,0x21F6,0x21F7,0x21F8

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprimimos el número de líneas en una tabla de caracteres con un tamaño de 7x16
   int total=(int)ExtShortArray.Size()/ROW_SIZE;
//--- en un ciclo a través de las filas de la tabla, generaremos en el registro 7 con
   for(int i=0; i<total; i++)
     {
      int row=i*ROW_SIZE;
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Arrow set %u: %s",i+1,ShortArrayToString(ExtShortArray,row,ROW_SIZE
     }
  /*
 resultado:
  Arrow set 1: ←↑→↓↔↕□□□□□□□□□□
  Arrow set 2: □□□□□□□□↕□□□□□□□
  Arrow set 3: □□□□□□□□□□□□□□□□
  Arrow set 4: □□□□□□□□□□□□□□□□
  Arrow set 5: □□⇒□⇔□□□□□□□□□□
  Arrow set 6: □□□□□□□□□□□□□□□□
  Arrow set 7: □□□□□□□□□□□□□□□□
  */
 }
```

Véase también

StringToShortArray, CharArrayToString, Uso de página de código

© 2000-2025, MetaQuotes Ltd.

TimeToString

Conversión del valor que contiene el tiempo en segundos transcurridos desde el 01.01.1970 a la cadena con el formato "yyyy.mm.dd hh:mi".

```
string  TimeToString(
   datetime  value,                          // número
   int       mode=TIME_DATE|TIME_MINUTES     // formato output
   );
```

Parámetros

value

[in] Tiempo en segundos de 00:00 1 de Enero de 1970.

mode=TIME_DATE|TIME_MINUTES

[in] Modo adicional del output de datos. Puede ser una bandera o bandera combinada:
TIME_DATE obtiene resultado en formato " yyyy.mm.dd " ,
TIME_MINUTES obtiene resultado en formato " hh:mi " ,
TIME_SECONDS obtiene resultado en formato " hh:mi:ss ".

Valor devuelto

Una cadena.

Ejemplo:

```
datetime ExtBarTimeOpen;

//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- establecemos el temporizador en un segundo
   EventSetTimer(1);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator deinitialization function                       |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   Comment("");
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
```

© 2000-2025, MetaQuotes Ltd.

```
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- obtenemos la hora de apertura de la barra actual
   ExtBarTimeOpen=time[rates_total-1];
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- anotamos la hora anterior de apertura de la barra
   static datetime bar_open_time=ExtBarTimeOpen;
//--- calculamos el número de segundos transcurridos desde la apertura de la barra
   static int seconds=int(TimeCurrent()-ExtBarTimeOpen);
//--- si la hora anterior de apertura de la barra no es igual a la actual, significa q
//--- anotamos la nueva hora de apertura como la anterior y ponemos los segundos a cer
   if(bar_open_time!=ExtBarTimeOpen)
     {
      bar_open_time=ExtBarTimeOpen;
      seconds=0;
     }
//--- aumentamos y corregimos el número de segundos transcurridos desde el momento de
   seconds++;
   if(seconds>PeriodSeconds(PERIOD_CURRENT))
      seconds=0;
//--- descripción de la hora de apertura de la barra en formato yyyy.mm.dd hh:mi
   string bar_time_open=TimeToString(ExtBarTimeOpen);
//--- descripción de la barra actual en formato yyyy.mm.dd hh:mi:ss
   string time_current=TimeToString(TimeCurrent(),TIME_DATE|TIME_MINUTES|TIME_SECONDS)
//--- número de segundos restante hasta la apertura de una nueva barra
   int    sec_left=PeriodSeconds(PERIOD_CURRENT)-seconds;
//--- descripción de la cantidad de tiempo restante hasta la apertura de una nueva bar
   string time_left=TimeToString(sec_left,TIME_MINUTES|TIME_SECONDS);
//--- creamos una línea de introducción
   string txt=StringFormat("Opening time of the current bar: %s\n"+
                           "Time Current: %s\n"+
                           "Seconds have passed since the bar opened: %d\n"+
                           "Approximately seconds left before bar closes: %d\n"+
                           "Time remaining until bar closes: %s",bar_time_open,time_cu
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos la hora de apertura de la barra, la hora actual en el comentario del g
//--- el número de segundos transcurridos desde la apertura de la barra actual y los c
//--- la hora hasta el cierre de la barra actual
   Comment(txt);
   /*
   resultado en M1:
   Opening time of the current bar: 2024.02.22 18:06
   Time Current: 2024.02.22 18:06:24
   Seconds have passed since the bar opened: 25
   Approximately seconds left before bar closes: 35
   Time remaining until bar closes: 00:00:35

   resultado en M5:
   Opening time of the current bar: 2024.02.22 18:05
   Time Current: 2024.02.22 18:07:28
   Seconds have passed since the bar opened: 149
   Approximately seconds left before bar closes: 151
   Time remaining until bar closes: 00:02:31

   resultado en H1:
   Opening time of the current bar: 2024.02.22 18:00
   Time Current: 2024.02.22 18:08:13
   Seconds have passed since the bar opened: 494
   Approximately seconds left before bar closes: 3106
   Time remaining until bar closes: 00:51:46

   resultado en D1:
   Opening time of the current bar: 2024.02.22 00:00
   Time Current: 2024.02.22 18:11:01
   Seconds have passed since the bar opened: 65462
   Approximately seconds left before bar closes: 20938
   Time remaining until bar closes: 05:48:58
   */
  }
```

Véase también

StringToTime, TimeToStruct

© 2000-2025, MetaQuotes Ltd.

NormalizeDouble

Redondeo del número con punto flotante hasta una precisión especificada.

```
double  NormalizeDouble(
   double  value,      // número normalizado
   int     digits      // cantidad de símbolos después del punto decimal
   );
```

Parámetros

value

[in] Valor con punto flotante.

digits

[in] Formato de precisión, número de dígitos después del punto decimal (0-8).

Valor devuelto

Valor del tipo double con una precisión especificada.

Nota

Valores calculados StopLoss, TakeProfit y los valores de precio de apertura de pedidos pendientes tienen que ser normalizados con la precisión cuyo valor puede ser obtenido por la función Digits().

Hay que tener en cuenta que cuando un número normalizado se visualiza en el Diario mediante Print(), éste puede tener una cantidad de dígitos tras la coma más grande de lo esperado. Por ejemplo,

```
   double a=76.671;              // número normalizado con 3 dígitos tras la coma
   Print("Print(76.671)=",a);    // vamos a mostrarlo tal como es
   Print("DoubleToString(a,8)=",DoubleToString(a,8)); // vamos a mostrarlo con una pre
```

mostrará en el terminal:

```
   DoubleToString(a,8)=76.67100000

   Print(76.671)=76.67100000000001
```

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
   double pi=M_PI;
   Print("pi = ",DoubleToString(pi,16));

   double pi_3=NormalizeDouble(M_PI,3);
   Print("NormalizeDouble(pi,3) = ",DoubleToString(pi_3,16))
   ;
   double pi_8=NormalizeDouble(M_PI,8);
   Print("NormalizeDouble(pi,8) = ",DoubleToString(pi_8,16));

   double pi_0=NormalizeDouble(M_PI,0);
   Print("NormalizeDouble(pi,0) = ",DoubleToString(pi_0,16));
/*
   Resultado:
   pi= 3.1415926535897931
   NormalizeDouble(pi,3)= 3.1419999999999999
   NormalizeDouble(pi,8)= 3.1415926499999998
   NormalizeDouble(pi,0)= 3.0000000000000000
*/
```

Véase también

DoubleToString, Tipos reales (double, float), Conversión de tipos

© 2000-2025, MetaQuotes Ltd.

StringToCharArray

Copia los caracteres de una cadena transformada de Unicode en ANSI en una parte indicada de un array del tipo uchar. La función devuelve el número de elementos copiados.

```
int  StringToCharArray(
   string  text_string,          // cadena de origen
   uchar&  array[],              // array
   int     start=0,              // posición de inicio en el array
   int     count=-1              // cantidad de símbolos
   uint    codepage=CP_ACP       // página de códigos
   );
```

Parámetros

text_string

[in] Cadena para el copiado.

array[]

[out] Array del tipo uchar.

start=0

[in] Posición de donde se empieza a copiar. Por defecto es 0.

count=-1

[in] Número de elementos del array para ser copiados. Determina la longitud de la cadena resultante. El valor por defecto es -1, esto significa el copiado hasta el final del array, o hasta el 0 de terminación. El 0 de terminación también va a ser copiado en el array de destino; en este caso, si hace falta, el tamaño del array dinámico puede ser aumentado hasta el tamaño de la cadena. Si el tamaño del array dinámico supera la longitud de la cadena, el tamaño del array no será reducido.

codepage=CP_ACP

[in] Valor de la página de códigos. Para algunas páginas de códigos más usadas están previstas constantes correspondientes.

Valor devuelto

Número de elementos copiados.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- string to convert
   string text = "This is a test of the StringToCharArray() function";

//--- convertimos la cadena de entrada en un array uchar según la página de códigos ir
   uchar char_array[];
```

© 2000-2025, MetaQuotes Ltd.

```
   int   copied=StringToCharArray(text, char_array);
   PrintFormat("String length: %u\nNumber of characters copied (with terminal 0): %d\r
               StringLen(text), copied, text);
//--- imprimimos en el registro el array obtenido
   ArrayPrint(char_array, 0, " | ");
   /*
  resultado:
   String length: 50
   Number of characters copied (with terminal 0): 51
   Array of characters for the string 'This is a test of the StringToCharArray() funct
   [ 0]  84 | 104 | 105 | 115 |  32 | 105 | 115 |  32 |  97 |  32 | 116 | 101 | 115 |
   [17]  32 | 116 | 104 | 101 |  32 |  83 | 116 | 114 | 105 | 110 | 103 |  84 | 111 |
   [34]  65 | 114 | 114 |  97 | 121 |  40 |  41 |  32 | 102 | 117 | 110 |  99 | 116 |
   */
  }
```

Véase también

CharArrayToString, StringToShortArray, Uso de página de código

© 2000-2025, MetaQuotes Ltd.

StringToColor

Convierte una cadena del tipo "R,G,B" o una cadena que contiene nombre de un color al valor del tipo color.

```
color  StringToColor(
   string  color_string      // representación literal de color
   );
```

Parámetros

color_string

[in] Representación literal de un color del tipo "R,G,B" o nombre de uno de los colores Web predefinidos.

Valor devuelto

Valor del color.

Ejemplo:

```
   color str_color=StringToColor("0,127,0");
   Print(str_color);
   Print((string)str_color);
//--- cambiamos un poco el color
   str_color=StringToColor("0,128,0");
   Print(str_color);
   Print((string)str_color);
```

Véase también

ColorToString, ColorToARGB

© 2000-2025, MetaQuotes Ltd.

StringToDouble

La función convierte la cadena que contiene la representación simbólica de un número al número del tipo double.

```
double  StringToDouble(
   string  value      // cadena
   );
```

Parámetros

value

[in] Cadena que contiene la representación simbólica de un número.

Valor devuelto

Valor del tipo double.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- string to convert
   string str = "12345.54321";
//--- convertir la cadena de entrada como un número real en una variable de tipo doubl
   double converted=StringToDouble(str);
//--- mostramos el número resultante en el registro con una precisión de 8 decimales.
   PrintFormat("The string '%s' is converted to the real number %.8f", str, converted)
   /*
  resultado:
   The string '12345.54321' is converted to the real number 12345.54321000
   */
  }
```

Véase también

NormalizeDouble, Tipos reales (double, float), Conversión de tipos

© 2000-2025, MetaQuotes Ltd.

StringToInteger

Conversión de una cadena que contiene la representación simbólica de un número al número del tipo long (entero).

```
long  StringToInteger(
   string  value      // Cadena
   );
```

Parámetros

value

[in] Cadena que contiene el número.

Valor devuelto

Valor del tipo long.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- string to convert
   string str = "12345.54321";
//--- convertimos la cadena de entrada como un número en una variable de tipo long
   long converted=StringToInteger(str);
//--- mostramos el número resultante en el registro con la parte fraccionaria cortada
   PrintFormat("The string '%s' is converted to the integer number %I64d", str, conver
   /*
  resultado:
   The string '12345.54321' is converted to the integer number 12345
   */
  }
```

Véase también

IntegerToString, Tipos reales (double, float), Conversión de tipos

© 2000-2025, MetaQuotes Ltd.

StringToShortArray

Copia una cadena símbolo por símbolo en una parte indicada de un array del tipo ushort. La función devuelve el número de elementos copiados.

```
int  StringToShortArray(
   string  text_string,      // cadena de origen
   ushort& array[],          // array
   int     start=0,          // posición de inicio en el array
   int     count=-1          // cantidad de símbolos
   );
```

Parámetros

text_string

[in] Cadena para el copiado.

array[]

[out] Array del tipo ushort (análogo del tipo wchar_t).

start=0

[in] Posición de donde se empieza a copiar. Por defecto es 0.

count=-1

[in] Número de elementos del array para ser copiados. Determina la longitud de la cadena resultante. El valor por defecto es -1, esto significa el copiado hasta el final del array, o hasta el 0 de terminación. El 0 de terminación también va a ser copiado en el array de destino; en este caso, si hace falta, el tamaño del array dinámico puede ser aumentado hasta el tamaño de la cadena. Si el tamaño del array dinámico supera la longitud de la cadena, el tamaño del array no será reducido.

Valor devuelto

Número de elementos copiados.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- string with chars to convert
   string text = "Chars: ♡♫☎✈☩☏❁✉☆☁♕♠®✧❄♣    ";
   ushort short_array[];
//--- buscamos la posición del carácter ":" en la cadena de entrada y extraemos la sub
   int    pos=StringFind(text, ":");
   string str=(pos<0 ? text : StringSubstr(text, pos+1));
//--- eliminamos de la cadena resultante los retornos de carro, los espacios y los car
   StringTrimLeft(str);
   StringTrimRight(str);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- copiamos la cadena resultante en un array ushort e imprimimos el array resultant
   int copied=StringToShortArray(str, short_array);
   PrintFormat("String of characters length: %u\n"
               "Number of characters copied (with terminal 0): %d\n"
               "Array of chars for the string '%s':",
               StringLen(str), copied, str);
   ArrayPrint(short_array, 0, " | ");
   /*
  resultado:
   String of characters length: 16
   Number of characters copied (with terminal 0): 17
   Array of chars for the string '♡♫☎✂✢☏⚙✉☆⌓♛♠®✧❄♣':
   10084 |  9835 |  9742 |  9992 | 10019 |  9743 |  9728 |  9993 |  9734 |  9729 |  98
   */
  }
```

Véase también

ShortArrayToString, StringToCharArray, Uso de página de código

© 2000-2025, MetaQuotes Ltd.

StringToTime

Conversión de una cadena que contiene una hora y/o fecha en el formato "yyyy.mm.dd [hh:mi]" a un número del tipo datetime.

```
datetime  StringToTime(
   const string  time_string      // cadena-fecha
   );
```

Parámetros

time_string

[in] Cadena en uno de los formatos indicados:

- "yyyy.mm.dd [hh:mi]"
- "yyyy.mm.dd [hh:mi:ss]"
- "yyyymmdd [hh:mi:ss]"
- "yyyymmdd [hhmiss]"
- "yyyy/mm/dd [hh:mi:ss]"
- "yyyy-mm-dd [hh:mi:ss]"

Valor retornado

Valor del tipo datetime, que contiene el número de segundos transcurridos desde el 01.01.1970.

Observación

Cualquier secuencia de símbolos de espacio y tabulación entre la fecha y la hora se considerará como un espacio, para que no sea necesario procesar adicionalmente la cadena *time_string* antes de llamar StringToTime().

Ejemplo:

```
//--- input parameters
input group    "The date can be entered in any of the formats:"
input group    "yyyy.mm.dd [hh:mi], yyyy.mm.dd [hh:mi:ss]"
input group    "yyyymmdd [hh:mi:ss], yyyymmdd [hhmiss]"
input group    "yyyy/mm/dd [hh:mi:ss], yyyy-mm-dd [hh:mi:ss]"
input string   InpDateStr;    // Please enter the date here as a string

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- convertimos el tiempo introducido en los parámetros de entrada como una cadena e
   datetime time=StringToTime(InpDateStr);
//--- mostramos en el registro la cadena introducida y la hora obtenida
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("Date entered as a string in the form '%s' is converted to datetime in
               InpDateStr, TimeToString(time, TIME_DATE|TIME_MINUTES|TIME_SECONDS));
//--- creamos una línea vertical en la fecha y hora resultante y desplazamos el gráfi
   if(CreateVLine(time))
      ChartNavigateToTime(time);
   /*
  resultado:
   Date entered as a string in the form '' is converted to datetime in the form '1970.
   Date entered as a string in the form '2024' is converted to datetime in the form '2
   Date entered as a string in the form '202400' is converted to datetime in the form
   Date entered as a string in the form '20240000' is converted to datetime in the for
   Date entered as a string in the form '2024022410' is converted to datetime in the
   Date entered as a string in the form '20240224 10' is converted to datetime in the
   Date entered as a string in the form '20240224 01' is converted to datetime in the
   Date entered as a string in the form '20240224 0030' is converted to datetime in th
   Date entered as a string in the form '20240224 0100' is converted to datetime in th
   */
  }
//+------------------------------------------------------------------+
//| Crea un objeto de línea vertical                                 |
//+------------------------------------------------------------------+
bool CreateVLine(const datetime line_time)
  {
   ResetLastError();

   string name=MQLInfoString(MQL_PROGRAM_NAME)+"_VLINE";
   if(!ObjectCreate(0, name, OBJ_VLINE, 0, line_time, 0))
     {
      Print("ObjectCreate() failed. Error code: ", GetLastError());
      return(false);
     }
   ObjectSetInteger(0, name, OBJPROP_STYLE, STYLE_DOT);
   ObjectSetInteger(0, name, OBJPROP_SELECTABLE, true);

   return(true);
  }
//+------------------------------------------------------------------+
//| Desplaza el gráfico a la hora indicada de apertura de la barra   |
//+------------------------------------------------------------------+
bool ChartNavigateToTime(const datetime time)
  {
   ChartSetInteger(0, CHART_AUTOSCROLL, false);
   ResetLastError();

   int bar=iBarShift(_Symbol, PERIOD_CURRENT, time);
   if(bar<0)
     {
      PrintFormat("%s: iBarShift() failed. Error code: %d", __FUNCTION__, GetLastError
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }

   long first=0;
   if(!ChartGetInteger(0, CHART_FIRST_VISIBLE_BAR, 0, first))
     {
      PrintFormat("%s: ChartGetInteger() failed. Error code: %d", __FUNCTION__, GetLas
      return(false);
     }

   return(ChartNavigate(0, CHART_CURRENT_POS, (int)first-bar));
  }
```

Mire también

TimeToString, TimeToStruct

© 2000-2025, MetaQuotes Ltd.

StringFormat

Formatea los parámetros obtenidos y devuelve una cadena.

```
string  StringFormat(
   string  format,      // cadena con descripción de formato
   ...     ...          // parámetros
   );
```

Parámetros

format

[in] Cadena que contiene el modo de formatear. Las reglas de formatear son las mismas que para la función PrintFormat

...

[in] Parámetros separados por coma.

Valor devuelto

Cadena.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
 {
//--- variables string
 string output_string;
 string temp_string;
 string format_string;
//--- preparamos encabezado de especificacion
 temp_string=StringFormat("Especificación del contrato en %s:\n",_Symbol);
 StringAdd(output_string,temp_string);
//--- mostrar valor int
 int digits=(int)SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
 temp_string=StringFormat(" SYMBOL_DIGITS = %d (número de dígitos tras la coma)\n",
 digits);
 StringAdd(output_string,temp_string);
//--- mostrar valor double con el número variable de dígitos tras la coma decimal
 double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
 format_string=StringFormat(" SYMBOL_POINT = %%.%df (valor de un punto)\n",
 digits);
 temp_string=StringFormat(format_string,point_value);
 StringAdd(output_string,temp_string);
//--- mostrar valor int
 int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
 temp_string=StringFormat(" SYMBOL_SPREAD = %d (spread actual en puntos)\n",
 spread);
 StringAdd(output_string,temp_string);
//--- mostrar valor int
 int min_stop=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
 temp_string=StringFormat(" SYMBOL_TRADE_STOPS_LEVEL = %d (sangría mínima en puntos pa
 min_stop);
 StringAdd(output_string,temp_string);
//--- mostrar valor double sin parte fraccionada
 double contract_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_CONTRACT_SIZE);
 temp_string=StringFormat(" SYMBOL_TRADE_CONTRACT_SIZE = %.f (tamaño del contrato)\n",
 contract_size);
 StringAdd(output_string,temp_string);
//--- mostrar valor double con la precisión predefinida
 double tick_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_TICK_SIZE);
 temp_string=StringFormat(" SYMBOL_TRADE_TICK_SIZE = %f (cambio mínimo del precio)\n",
 tick_size);
 StringAdd(output_string,temp_string);
//--- determinar el modo para calcular los swaps
 int swap_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_SWAP_MODE);
 string str_swap_mode;
 switch(swap_mode)
 {
 case SYMBOL_SWAP_MODE_DISABLED: str_swap_mode="SYMBOL_SWAP_MODE_DISABLED (no hay swap
 case SYMBOL_SWAP_MODE_POINTS: str_swap_mode="SYMBOL_SWAP_MODE_POINTS (en puntos)"; br
 case SYMBOL_SWAP_MODE_CURRENCY_SYMBOL: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_SYMBO
 case SYMBOL_SWAP_MODE_CURRENCY_MARGIN: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_MARGI
 case SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_DEPO
 case SYMBOL_SWAP_MODE_INTEREST_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST_CURR
 case SYMBOL_SWAP_MODE_INTEREST_OPEN: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST_OPEN (e
 case SYMBOL_SWAP_MODE_REOPEN_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_CURRENT
 case SYMBOL_SWAP_MODE_REOPEN_BID: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_BID (reapert
 }
//--- mostrar valor string
 temp_string=StringFormat(" SYMBOL_SWAP_MODE = %s\n",
 str_swap_mode);
 StringAdd(output_string,temp_string);
//--- mostrar valor double con la precisión predefinida
 double swap_long=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_LONG);
```

© 2000-2025, MetaQuotes Ltd.

```
 temp_string=StringFormat(" SYMBOL_SWAP_LONG = %f (swap de compra)\n",
 swap_long);
 StringAdd(output_string,temp_string);
//--- mostrar valor double con precisión predefinida
 double swap_short=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_SHORT);
 temp_string=StringFormat(" SYMBOL_SWAP_SHORT = %f (swap de venta)\n",
 swap_short);
 StringAdd(output_string,temp_string);
//--- determinar el modo de trading
 int trade_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_MODE);
 string str_trade_mode;
 switch(trade_mode)
 {
 case SYMBOL_TRADE_MODE_DISABLED: str_trade_mode="SYMBOL_TRADE_MODE_DISABLED (está pro
 case SYMBOL_TRADE_MODE_LONGONLY: str_trade_mode="SYMBOL_TRADE_MODE_LONGONLY (están pe
 case SYMBOL_TRADE_MODE_SHORTONLY: str_trade_mode="SYMBOL_TRADE_MODE_SHORTONLY (están
 case SYMBOL_TRADE_MODE_CLOSEONLY: str_trade_mode="SYMBOL_TRADE_MODE_CLOSEONLY (están
 case SYMBOL_TRADE_MODE_FULL: str_trade_mode="SYMBOL_TRADE_MODE_FULL (no hay limitacio
 }
//--- mostrar valor string
 temp_string=StringFormat(" SYMBOL_TRADE_MODE = %s\n",
 str_trade_mode);
 StringAdd(output_string,temp_string);
//--- mostrar valor double en modo compacto
 double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
 temp_string=StringFormat(" SYMBOL_VOLUME_MIN = %g (volumen mínimo de la transacción)\
 StringAdd(output_string,temp_string);
//--- mostrar valor double en modo compacto
 double volume_step=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_STEP);
 temp_string=StringFormat(" SYMBOL_VOLUME_STEP = %g (paso mínimo del cambio del volume
 StringAdd(output_string,temp_string);
//--- mostrar valor double en modo compacto
 double volume_max=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MAX);
 temp_string=StringFormat(" SYMBOL_VOLUME_MAX = %g (volumen máximo de la transacción)\
 StringAdd(output_string,temp_string);
//--- determinar el modo del cálculo de fondos del margen
 int calc_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_CALC_MODE);
 string str_calc_mode;
 switch(calc_mode)
 {
 case SYMBOL_CALC_MODE_FOREX:str_calc_mode="SYMBOL_CALC_MODE_FOREX (Forex)";break;
 case SYMBOL_CALC_MODE_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_FUTURES (futuros)";brea
 case SYMBOL_CALC_MODE_CFD:str_calc_mode="SYMBOL_CALC_MODE_CFD (CFD)";break;
 case SYMBOL_CALC_MODE_CFDINDEX:str_calc_mode="SYMBOL_CALC_MODE_CFDINDEX (CFD de índic
 case SYMBOL_CALC_MODE_CFDLEVERAGE:str_calc_mode="SYMBOL_CALC_MODE_CFDLEVERAGE (CFD pa
 case SYMBOL_CALC_MODE_EXCH_STOCKS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_STOCKS (tradi
 case SYMBOL_CALC_MODE_EXCH_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FUTURES (trad
 case SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FUTURES
 }
//--- mostra valor string
 temp_string=StringFormat(" SYMBOL_TRADE_CALC_MODE = %s\n",
 str_calc_mode);
 StringAdd(output_string,temp_string);
//--- mostrar valor double con 2 dígitos tras la coma decimal
 double margin_initial=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_INITIAL);
 temp_string=StringFormat(" SYMBOL_MARGIN_INITIAL = %.2f (margen inicial)\n",
 margin_initial);
 StringAdd(output_string,temp_string);
//--- mostrar valor double con 2 dígitos tras la coma decimal
 double margin_maintenance=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_MAINTENANCE);
 temp_string=StringFormat(" SYMBOL_MARGIN_MAINTENANCE = %.2f (margen de soporte)\n",
```

© 2000-2025, MetaQuotes Ltd.

```
 margin_maintenance);
 StringAdd(output_string,temp_string);
//--- mostrar valor int
 int freeze_level=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_FREEZE_LEVEL);
 temp_string=StringFormat(" SYMBOL_TRADE_FREEZE_LEVEL = %d (distancia de congelación d
 freeze_level);
 StringAdd(output_string,temp_string);
 Print(output_string);
 Comment(output_string);
/* resultado de ejecución
 Especificación del contrato para EURUSD:
 SYMBOL_DIGITS = 5 (número de dígitos tras la coma)
 SYMBOL_POINT = 0.00001 (valor de un punto)
 SYMBOL_SPREAD = 10 (spread actual en puntos)
 SYMBOL_TRADE_STOPS_LEVEL = 18 (sangría mínima en puntos para las órdenes stop)
 SYMBOL_TRADE_CONTRACT_SIZE = 100000 (tamaño del contrato)
 SYMBOL_TRADE_TICK_SIZE = 0.000010 (cambio mínimo del precio)
 SYMBOL_SWAP_MODE = SYMBOL_SWAP_MODE_POINTS (en puntos)
 SYMBOL_SWAP_LONG = -0.700000 (swap de compra)
 SYMBOL_SWAP_SHORT = -1.000000 (swap de venta)
 SYMBOL_TRADE_MODE = SYMBOL_TRADE_MODE_FULL (no hay limitaciones para las operaciones
 SYMBOL_VOLUME_MIN = 0.01 (valor mínimo de la transacción)
 SYMBOL_VOLUME_STEP = 0.01 (paso mínimo del cambio del volumen de la transacción)
 SYMBOL_VOLUME_MAX = 500 (valor máximo de la transacción)
 SYMBOL_TRADE_CALC_MODE = SYMBOL_CALC_MODE_FOREX (Forex)
 SYMBOL_MARGIN_INITIAL = 0.00 (margen inicial)
 SYMBOL_MARGIN_MAINTENANCE = 0.00 (margen de soporte)
 SYMBOL_TRADE_FREEZE_LEVEL = 0 (distancia de congelación de las operaciones en puntos)
*/
 }
```

Véase también

PrintFormat, DoubleToString,ColorToString, TimeToString

© 2000-2025, MetaQuotes Ltd.

Funciones matemáticas

Conjunto de funciones matemáticas y trigonométricas.

Las funciones matemáticas se diseñaron originalmente para realizar operaciones matemáticas con cantidades escalares. Ahora la mayoría de estas funciones pueden utilizarse con los nuevos tipos de datos (matrices y vectores) MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathCosh, MathSinh, MathTanh. En este caso, la matriz o el vector se procesarán término a término. Ejemplo:

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

En el caso de MathMod y MathPow, el segundo parámetro podrá ser un escalar, o una matriz o vector del tamaño correspondiente.

Función	Acción
MathAbs	Devuelve el valor absoluto (modular) de un número que se le ha pasado
MathArccos	Devuelve el valor de arcocoseno de (x) en radianes
MathArcsin	Devuelve el valor de arcseno de (x) en radianes
MathArctan	Devuelve el valor de arcotangente de (x) en radianes
MathArctan2	Devuelve en radianes el valor del ángulo cuya tangente es igual al cociente de los dos números especificados.
MathClassify	Devuelve el tipo de un número real
MathCeil	Devuelve el valor numérico entero más cercano desde arriba
MathCos	Devuelve el coseno de un número
MathExp	Devuelve el exponente de un número
MathFloor	Devuelve el valor numérico entero más cercano desde abajo
MathLog	Devuelve un logaritmo neperiano (natural)
MathLog10	Devuelve el logaritmo de un número en base 10

© 2000-2025, MetaQuotes Ltd.

Función	Acción
MathMax	Devuelve el valor máximo de dos valores numéricos
MathMin	Devuelve el valor mínimo de dos valores numéricos
MathMod	Devuelve el resto real de la división de dos números
MathPow	Eleva la base a la potencia indicada
MathRand	Devuelve un número pseudoaleatorio en el rango de 0 a 32767
MathRound	Redondea un número hasta el entero más cercano
MathSin	Devuelve el seno de un número
MathSqrt	Devuelve una raíz cuadrada
MathSrand	Define el estado inicial del generador pseudoaleatorio de números enteros
MathTan	Devuelve la tangente de un número
MathIsValidNumber	Verifica la correctitud de un número real
MathExpm1	Devuelve el valor de la expresión MathExp(x)-1
MathLog1p	Devuelve el valor de la expresión MathLog(1+x)
MathArccosh	Devuelve el valor del arcocoseno hiperbólico
MathArcsinh	Devuelve el valor del arcoseno hiperbólico
MathArctanh	Devuelve el valor de la arcotangente hiperbólica
MathCosh	Devuelve el coseno hiperbólico
MathSinh	Devuelve el seno hiperbólico
MathTanh	Devuelve la tangente hiperbólica
MathSwap	Cambia el orden de los bytes en un valor del tipo ushort/uint/ushort
MathSwap	Change the order of bytes in the ushort/uint/ushort types value

© 2000-2025, MetaQuotes Ltd.

MathAbs

La función devuelve el valor absoluto (modular) de un número que se le ha pasado.

```
double  MathAbs(
   double  value      // número
   );
```

Parámetros

value

[in] Valor numérico.

Valor devuelto

Valor del tipo double, más o igual a cero.

Nota

En vez de la función MathAbs() se puede usar la función fabs().

Ejemplo:

```
//--- input parameters
input int      InpValue = -10;   // Enter any value here

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el valor absoluto del número introducido en los parámetros de entrada
   uint abs_value=MathAbs(InpValue);
//--- imprimimos los valores en el diario de registro
   PrintFormat("The entered value %d received the value %u after using the MathAbs() f
  }
```

© 2000-2025, MetaQuotes Ltd.

MathArccos

Devuelve el valor de arcocoseno de (x) en el rango de 0 a π en radianes.

```
double  MathArccos(
   double  val     // -1<val<1
   );
```

Parámetros

val

[in] Valor val entre -1 y 1, cuyo arcocoseno tiene que ser calculado.

Valor devuelto

Arcocoseno de un número en radianes. Si val es menos de -1 o más de 1, la función devuelve NaN (valor indeterminado).

Nota

En vez de la función MathArccos() se puede usar la función acos().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/1000);
   delta[0]=-1;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor del arcocoseno para cada valor del vector X
   vector Y=MathArccos(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

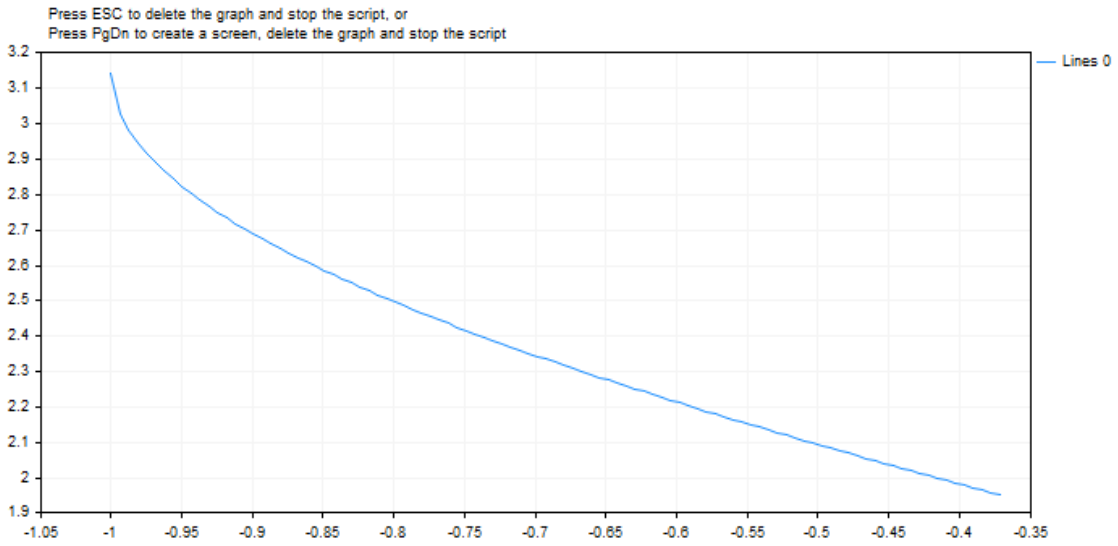
© 2000-2025, MetaQuotes Ltd.

```
//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathArcsin

Devuelve arcseno (x) en el rango de -π/2 a π/2 radianes.

```
double  MathArcsin(
   double  val      // -1<value<1
   );
```

Parámetros

val

[in] Valor val entre -1 y 1, cuyo arcseno tiene que ser calculado.

Valor devuelto

Arcseno de un número val en radianes en el rango de -π/2 a π/2 radianes. Si val es menos de -1 o más de 1, la función devuelve NaN (valor indeterminado).

Nota

En vez de la función MathArcsin() se puede usar la función asin().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/1000);
   delta[0]=-1;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor del seno para cada valor del vector X
   vector Y=MathArcsin(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

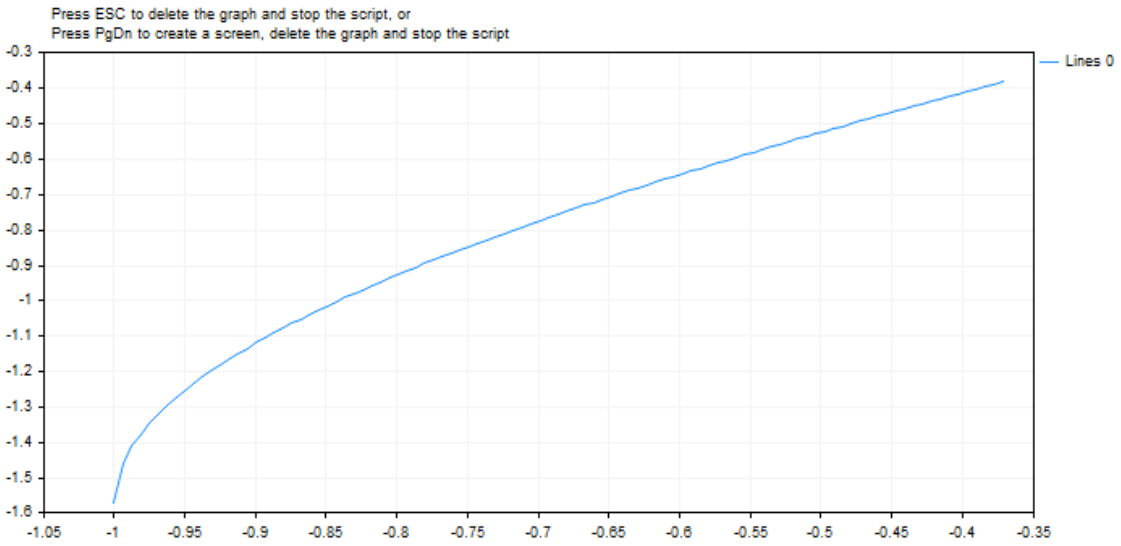
© 2000-2025, MetaQuotes Ltd.

```
//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRIT
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathArctan

Devuelve arcotangente de (x). Si x es igual a 0, la función devuelve 0.

```
double  MathArctan(
   double  value      // tangente
   );
```

Parámetros

value

[in] Número que representa tangente.

Valor devuelto

MathArctan devuelve un valor en el rango de -π/2 a π/2 radianes.

Nota

En vez de la función MathArctan() se puede usar la función atan().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor del arcotangente para cada valor del vector X
   vector Y=MathArctan(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
```

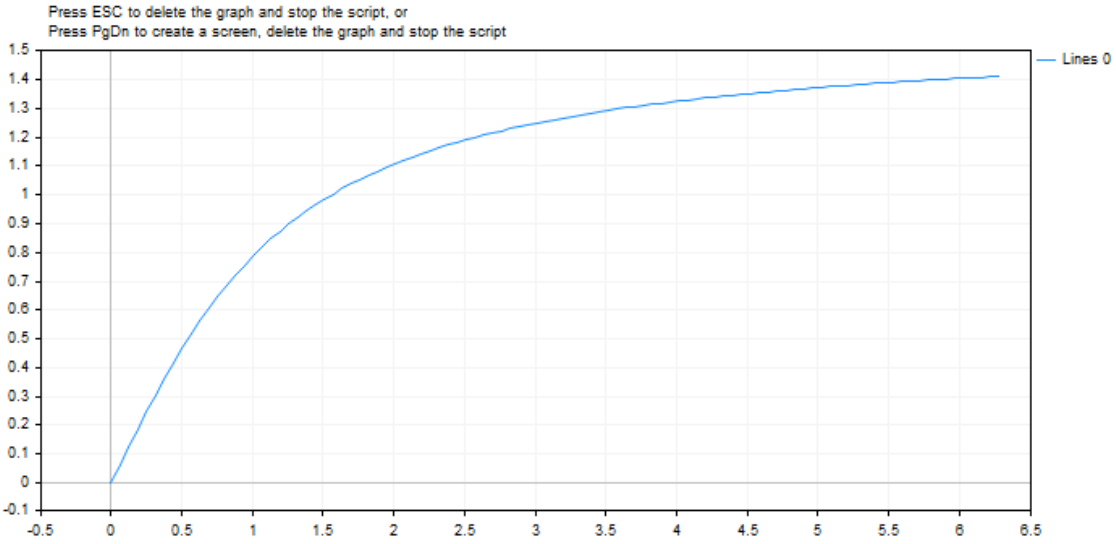
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathArctan2

Devuelve en radianes el valor del ángulo cuya tangente es igual al cociente de los dos números especificados.

```
double  MathArctan2(
   double  y      // coordenada Y del punto
   double  x      // coordenada X del punto
   );
```

Parámetros

y

[in] Número que representa la coordenada Y.

x

[in] Número que representa la coordenada X.

Valor retornado

MathArctan2 devuelve el valor del ángulo θ en un intervalo de π a π radianes, así que MathTan(θ) =y/x.

Preste atención a lo siguiente:

- Para (x, y) en el cuadrante 1, 0 < θ < π/2
- Para (x, y) en el cuadrante 2, π/2 < θ≤π
- Para (x, y) en el cuadrante 3, -π < θ < -π/2
- Para (x, y) en el cuadrante 4, -π/2 < θ < 0

Más abajo se muestra el valor devuelto para los puntos fuera de los cuadrantes indicados:

- Si Y es igual a 0, y X no es negativo, entonces θ = 0.
- Si Y es igual a 0, y X es negativo, entonces θ = π.
- Si Y es un número positivo, y X es igual a 0, entonces θ = π/2.
- Si Y es un número negativo, y X es igual a 0, entonces θ = -π/2.
- Si Y es igual 0, y X es igual a 0, entonces θ = 0.

Observación

En lugar de la función MathArctan2(), se puede utilizar la función atan2().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

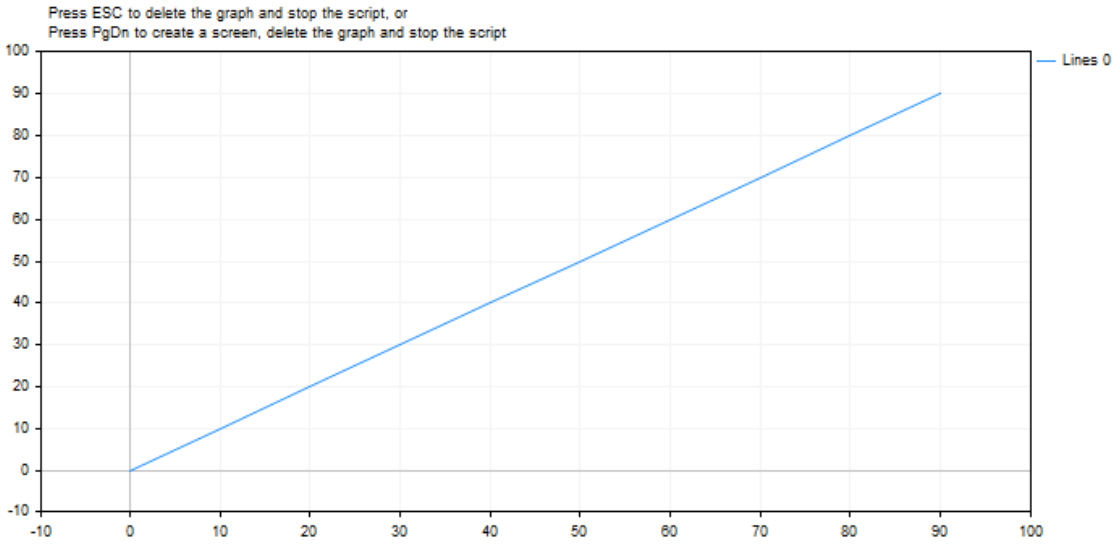
```
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(10,10);
   delta[0]=0;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor del arcotangente para cada valor del vector X
   vector Y=delta.CumSum();

   Print("vector delta = \n",delta);
   Print("vector X = \n",X);
   Print("vector Y = \n",Y);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[];;
   double y_array[];;
   X.Swap(x_array);
   Y.Swap(y_array);

   double array[10];
   for(int i=0; i<10; i++)
     {
      array[i]=MathArctan2(y_array[i],x_array[i]);
     }

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
```

© 2000-2025, MetaQuotes Ltd.

```
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:

© 2000-2025, MetaQuotes Ltd.



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathClassify

Determina el tipo de un número real y retorna el resultado en forma de valor de la enumeración ENUM_FP_CLASS

```
ENUM_FP_CLASS  MathClassify(
   double  value      // número real
   );
```

Parámetros

value

[in] Número real comprobado

Valor retornado

Valor de la enumeración ENUM_FP_CLASS

ENUM_FP_CLASS

Identificador	Descripción
FP_SUBNORMAL	Número subnormal que se encuentra más próximo a cero que el número normalizado DBL_MIN menor representable (2.2250738585072014e-308)
FP_NORMAL	Número normalizado que se encuentra en el intervalo de 2.2250738585072014e-308 a 1.7976931348623158e+308
FP_ZERO	Cero positivo o negativo
FP_INFINITE	Un número que no puede representarse con el tipo correspondiente, infinito positivo o negativo
FP_NAN	No es un número

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- test NaN
  double nan=double("nan");
  PrintFormat("Test NaN: %G is %s, MathIsValidNumber(NaN)=%s",
              nan,
              EnumToString(MathClassify(nan)),
              (string)MathIsValidNumber(nan));
//--- test infinity
```

© 2000-2025, MetaQuotes Ltd.

```
  double inf=double("inf");
  PrintFormat("Test Inf: %G is %s, MathIsValidNumber(inf)=%s",
              inf,
              EnumToString(MathClassify(inf)),
              (string)MathIsValidNumber(inf));
//--- test normal value
  double normal=1.2345e6;
  PrintFormat("Test Normal: %G is %s, MathIsValidNumber(normal)=%s",
              normal,
              EnumToString(MathClassify(normal)),
              (string)MathIsValidNumber(normal));
//--- test subnormal value
  double sub_normal=DBL_MIN/2.0;
  PrintFormat("Test Subnormal: %G is %s, MathIsValidNumber(sub_normal)=%s",
              sub_normal,
              EnumToString(MathClassify(sub_normal)),
              (string)MathIsValidNumber(sub_normal));
//--- test zero value
  double zero=0.0/(-1);
  PrintFormat("Test Zero: %G is %s, MathIsValidNumber(zero)=%s",
              zero,
              EnumToString(MathClassify(zero)),
              (string)MathIsValidNumber(zero));
 }
 /*
 Result:
   Test NaN: NAN is FP_NAN, MathIsValidNumber(NaN)=false
   Test Inf: INF is FP_INFINITE, MathIsValidNumber(inf)=false
   Test Normal: 1.2345E+06 is FP_NORMAL, MathIsValidNumber(normal)=true
   Test Subnormal: 1.11254E-308 is FP_SUBNORMAL, MathIsValidNumber(sub_normal)=true
   Test Zero: -0 is FP_ZERO, MathIsValidNumber(zero)=true
*/
//+------------------------------------------------------------------+
```

Ver también

Tipos reales (double, float), MathIsValidNumber

© 2000-2025, MetaQuotes Ltd.

MathCeil

Devuelve el valor numérico entero más cercano desde arriba.

```
double  MathCeil(
   double  val      // número
   );
```

Parámetros

val

[in] Valor numérico.

Valor devuelto

Valor numérico que representa el número entero más pequeño que supera o equivale a val.

Nota

En vez de la función MathCeil() se puede usar la función ceil().

Ejemplo:

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos las variables para la transformación
   double value=0;                     // número real para la transformación MathCeil
   int    ceil_value=0;                // aquí obtenemos el resultado
//--- en un ciclo por el número de incrementos decimales de un número real
   for(int i=0; i<VALUES_TOTAL; i++)
     {
      //--- aumentamos el valor del número,
      //--- obtenemos el valor entero más próximo por arriba
      //--- y mostramos en el diario de registro los valores de control
      value+=0.1;
      ceil_value=(int)MathCeil(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, ceil value: %d",value,ceil_value);
      /*
     resultado:
      value: 0.1, ceil value: 1
      value: 0.2, ceil value: 1
      value: 0.3, ceil value: 1
      value: 0.4, ceil value: 1
      value: 0.5, ceil value: 1
      value: 0.6, ceil value: 1
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, ceil value: 1
      value: 0.8, ceil value: 1
      value: 0.9, ceil value: 1
      value: 1.0, ceil value: 1
      value: 1.1, ceil value: 2
      value: 1.2, ceil value: 2
      value: 1.3, ceil value: 2
      value: 1.4, ceil value: 2
      value: 1.5, ceil value: 2
      value: 1.6, ceil value: 2
      value: 1.7, ceil value: 2
      value: 1.8, ceil value: 2
      value: 1.9, ceil value: 2
      value: 2.0, ceil value: 2
      value: 2.1, ceil value: 3
      value: 2.2, ceil value: 3
      value: 2.3, ceil value: 3
      value: 2.4, ceil value: 3
      value: 2.5, ceil value: 3
      value: 2.6, ceil value: 3
      value: 2.7, ceil value: 3
      value: 2.8, ceil value: 3
      value: 2.9, ceil value: 3
      value: 3.0, ceil value: 3
      value: 3.1, ceil value: 4
      */
    }
}
```

© 2000-2025, MetaQuotes Ltd.

MathCos

La función devuelve el coseno de un ángulo.

```
double  MathCos(
   double  value      // número
   );
```

Parámetros

value

[in] Ángulo en radianes.

Valor devuelto

Valor del tipo double en el rango de -1 a 1.

Nota

En vez de la función MathCos() se puede usar la función cos().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor del coseno para cada valor del vector X
   vector Y=MathCos(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
```

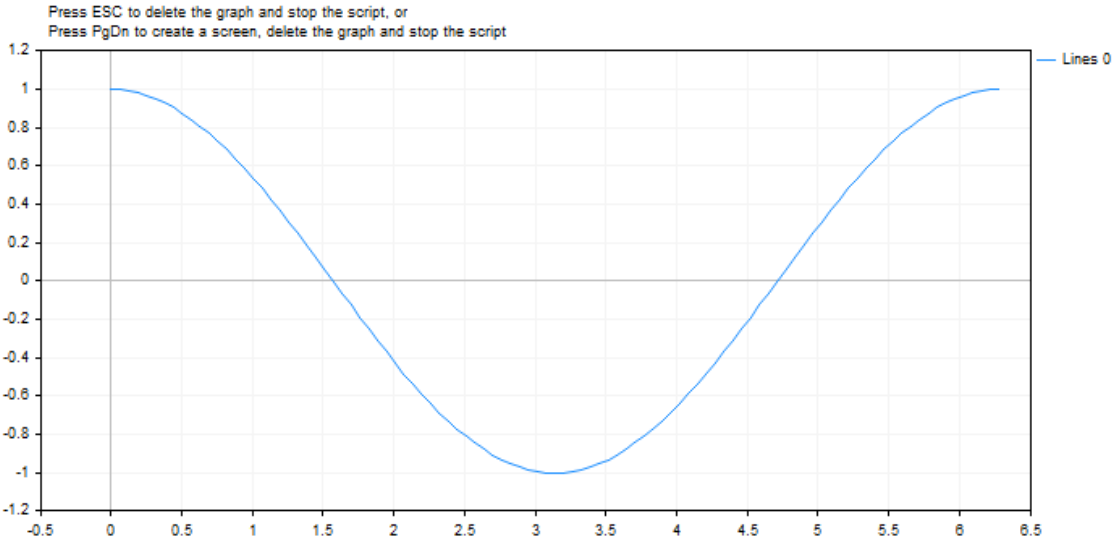
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathExp

La función devuelve el valor del número e elevado a la potencia d.

```
double  MathExp(
   double  value      // potencia para el número e
   );
```

Parámetros

value

[in] Número que especifica la potencia.

Valor devuelto

Número del tipo double. Al superar el límite la función devuelve INF (infinito), en caso de perder el orden MathExp devolverá 0.

Nota

En vez de la función MathExp() se puede usar la función exp().

Véase también

Tipos reales (double, float)

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos 9 valores de 0 a 8 con paso 1
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- calculamos el valor del número "e" (número de Euler) en el grado de cada valor d
   X=MathExp(X);
   Print("MathExp(X) = \n",X);

//--- transferimos los valores calculados de vector a array
   double y_array[];
   X.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
```

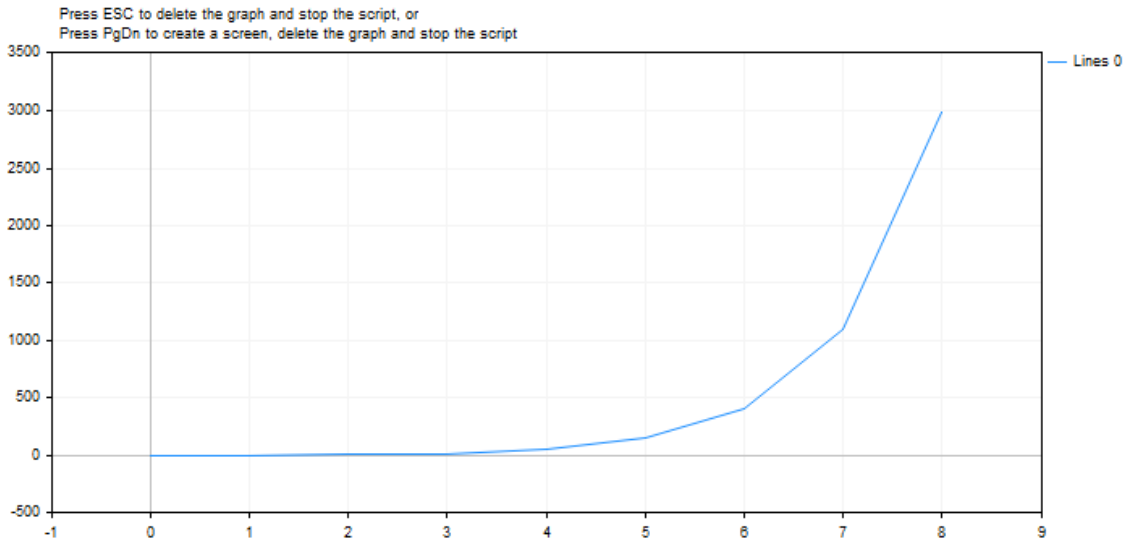
© 2000-2025, MetaQuotes Ltd.

```
   CurvePlot(y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Rellena el vector con el valor value con el incremento step      |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRIT
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathFloor

Devuelve el valor numérico entero más cercano desde abajo.

```
double  MathFloor(
   double  val     // número
   );
```

Parámetros

val

[in] Valor numérico.

Valor devuelto

Valor numérico que representa el número entero más grande, que es menos o igual a val.

Nota

En vez de la función MathFloor() se puede usar la función floor().

Ejemplo:

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos las variables para la transformación
   double value=0;                     // número real para la transformación MathFloor
   int    floor_value=0;               // aquí obtenemos el resultado
//--- en un ciclo por el número de incrementos decimales de un número real
   for(int i=0; i<VALUES_TOTAL; i++)
     {
      //--- aumentamos el valor del número,
      //--- obtenemos el valor entero más próximo por abajo
      //--- y mostramos en el diario de registro los valores de control
      value+=0.1;
      floor_value=(int)MathFloor(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, floor value: %d",value,floor_value);
      /*
     resultado:
      value: 0.1, floor value: 0
      value: 0.2, floor value: 0
      value: 0.3, floor value: 0
      value: 0.4, floor value: 0
      value: 0.5, floor value: 0
      value: 0.6, floor value: 0
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, floor value: 0
      value: 0.8, floor value: 0
      value: 0.9, floor value: 0
      value: 1.0, floor value: 1
      value: 1.1, floor value: 1
      value: 1.2, floor value: 1
      value: 1.3, floor value: 1
      value: 1.4, floor value: 1
      value: 1.5, floor value: 1
      value: 1.6, floor value: 1
      value: 1.7, floor value: 1
      value: 1.8, floor value: 1
      value: 1.9, floor value: 1
      value: 2.0, floor value: 2
      value: 2.1, floor value: 2
      value: 2.2, floor value: 2
      value: 2.3, floor value: 2
      value: 2.4, floor value: 2
      value: 2.5, floor value: 2
      value: 2.6, floor value: 2
      value: 2.7, floor value: 2
      value: 2.8, floor value: 2
      value: 2.9, floor value: 2
      value: 3.0, floor value: 3
      value: 3.1, floor value: 3
      */
     }
```

© 2000-2025, MetaQuotes Ltd.

MathLog

Devuelve un logaritmo neperiano (natural).

```
double  MathLog(
   double  val      // número para coger el logaritmo
   );
```

Parámetros

val

[in] Valor cuyo logaritmo tiene que ser calculado.

Valor devuelto

En caso de éxito devuelve el logaritmo natural de val. Si el valor de val es negativo, la función devuelve NaN (valor indeterminado). Si val es igual a 0, la función devuelve INF (infinito).

Nota

En vez de la función MathLog() se puede usar la función log().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos 9 valores de 0 a 8 con paso 1
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- calculamos el logaritmo para cada valor del vector X
   X=MathLog(X);
   Print("MathLog(X) = \n",X);

//--- transferimos los valores calculados de vector a array
   double y_array[];
   X.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
```

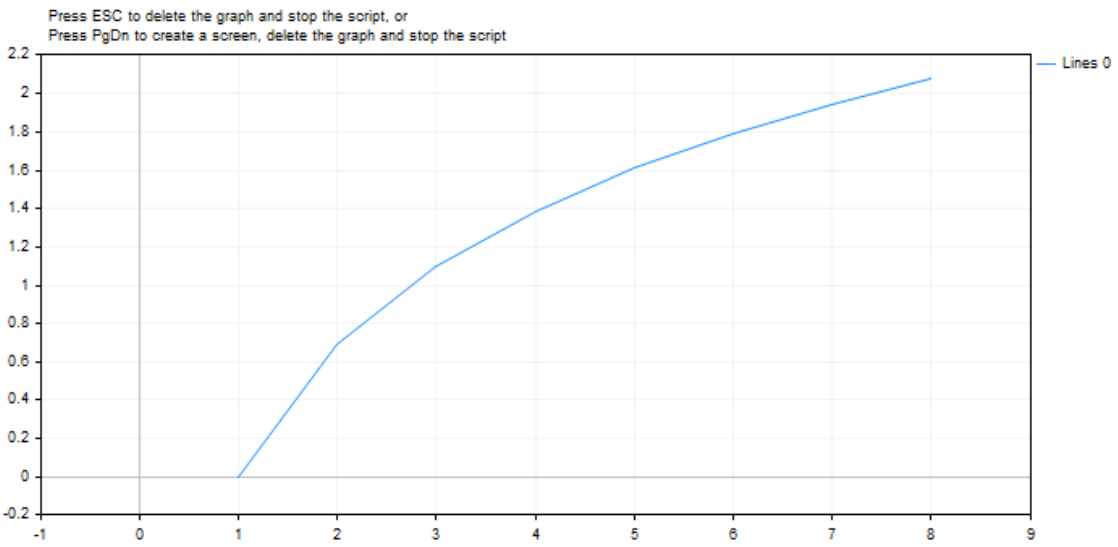
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
   /*
  resultado:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog(X) =
   [-inf,0,0.6931471805599453,1.09861228866811,1.386294361119891,1.6094379124341,1.791
   */
  }
//+------------------------------------------------------------------+
//| Rellena el vector con el valor value con el incremento step      |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQLI
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



© 2000-2025, MetaQuotes Ltd.

Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathLog

Devuelve el logaritmo de un número en base 10.

```
double  MathLog10(
   double  val      // número para coger el logaritmo
   );
```

Parámetros

val

[in] Valor cuyo logaritmo decimal tiene que ser calculado.

Valor devuelto

En caso de éxito devuelve el logaritmo decimal de val. Si el valor de val es negativo, la función devuelve NaN (valor indeterminado). Si val es igual a 0, la función devuelve INF (infinito).

Nota

En vez de la función MathLog10() se puede usar la función log10().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos 9 valores de 0 a 8 con paso 1
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- calculamos el logaritmo decimal para cada valor del vector X
   X=MathLog10(X);
   Print("MathLog10(X) = \n",X);

//--- transferimos los valores calculados de vector a array
   double y_array[];
   X.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
```

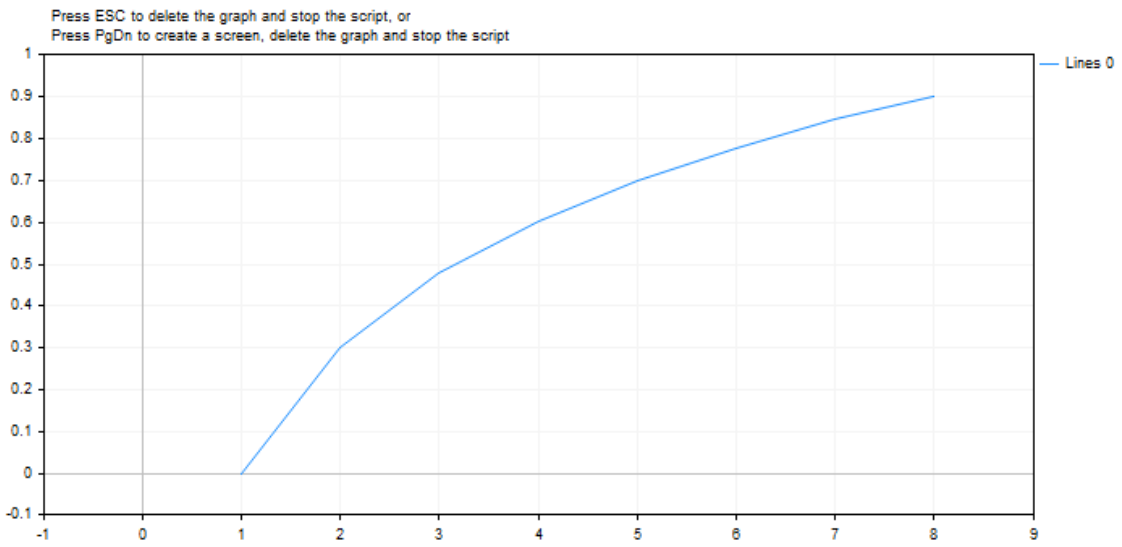
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
   /*
  resultado:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog10(X) =
   [-inf,0,0.3010299956639812,0.4771212547196624,0.6020599913279624,0.6989700043360189
   */
  }
//+------------------------------------------------------------------+
//| Rellena el vector con el valor value con el incremento step      |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQLI
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



© 2000-2025, MetaQuotes Ltd.

Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathMax

La función devuelve el valor máximo de dos valores numéricos.

```
double  MathMax(
   double  value1,      // primer número
   double  value2       // segundo número
   );
```

Parámetros

value1

[in] Primer valor numérico.

value2

[in] Segundo valor numérico.

Valor devuelto

El número más grande de los dos.

Nota

En vez de la función MathMax() se puede usar la función fmax(). Las funciones fmax(), fmin(), MathMax(), MathMin() pueden trabajar con tipos enteros sin conversión al tipo double.

Si los parámetros de diferentes tipos se pasan a la función, el parámetro del tipo menor automáticamente se convierte al tipo mayor. El tipo de valor devuelto corresponde al tipo mayor.

Si se pasan los datos del mismo tipo, no se realiza ninguna conversión.

Ejemplo:

```
//--- input parameters
input int                  InpPeriod = 10;            // Periodo de cálculo de la medi
input ENUM_MA_METHOD       InpMethod = MODE_SMA;      // Método de cálculo de la media
input ENUM_APPLIED_PRICE   InpPrice  = PRICE_CLOSE;   // Precio de cálculo de la media

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- si para el periodo de la media móvil se ha establecido un valor inferior a 1, se
   int period=(InpPeriod<1 ? 10 : InpPeriod);
//--- creamos el manejador del indicador Moving Average
   int handle=iMA(Symbol(),Period(),period,0,InpMethod,InpPrice);
   if(handle==INVALID_HANDLE)
     {
      Print("No se ha podido crear el manejador del indicador Moving Average. Error ",
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- obtenemos el precio Bid actual
   double bid=0;
   ResetLastError();
   if(!SymbolInfoDouble(Symbol(),SYMBOL_BID,bid))
     {
      Print("No se ha podido obtener el precio Bid. Error ",GetLastError());
      return;
     }
//--- obtenemos el valor de la media móvil en la barra actual
   double array[1];
   int    copied=CopyBuffer(handle,0,0,1,array);
   if(copied!=1)
     {
      Print("No se ha podido obtener los datos de Moving Average. Error ",GetLastErro
      return;
     }
//--- obtenemos el precio más alto de los dos (precio Bid y valor de Moving Average) y
   double max_price=MathMax(bid,array[0]);
   PrintFormat("Bid: %.*f, Moving Average: %.*f, es el mayor precio de los dos: %.*f",
   PrintFormat("El precio Bid %s de la media móvil",(bid>array[0] ? "más alto" : bid<a
  }
```

© 2000-2025, MetaQuotes Ltd.

MathMin

La función devuelve el valor mínimo de dos valores numéricos.

```
double  MathMin(
   double  value1,      // primer número
   double  value2       // segundo número
   );
```

Parámetros

value1

[in] Primer valor numérico.

value2

[in] Segundo valor numérico.

Valor devuelto

El número más pequeño de los dos.

Nota

En vez de la función MathMin() se puede usar la función fmin(). Las funciones fmax(), fmin(), MathMax(), MathMin() pueden trabajar con tipos enteros sin conversión al tipo double.

Si los parámetros de diferentes tipos se pasan a la función, el parámetro del tipo menor automáticamente se convierte al tipo mayor. El tipo de valor devuelto corresponde al tipo mayor.

Si se pasan los datos del mismo tipo, no se realiza ninguna conversión.

Ejemplo:

```
//--- input parameters
input uint  InpBars  =  100000;  // Número deseado de barras

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el número de barras disponibles en el servidor
   uint bars_total = Bars(Symbol(),Period());
   if(bars_total==0)
     {
      PrintFormat("Los datos de la serie temporal %s %s %s aún no se han generado en e
      return;
     }
//--- obtenemos el número mínimo de barras a partir de dos valores: de las barras disp
   int bars = (int)MathMin(bars_total,InpBars);
//--- si se solicitan más barras de las que están disponibles en el servidor, lo indic
```

© 2000-2025, MetaQuotes Ltd.

```
   if(bars_total<InpBars)
      PrintFormat("El número de barras en el servidor (%u) es inferior al solicitado
//--- mostramos en el diario de registro el número de barras disponibles para trabajar
   Print("Barras disponibles para el trabajo: ",bars);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathMod

Devuelve el resto real de la división de dos números.

```
double  MathMod(
   double  value,       // dividiendo
   double  value2       // divisor
   );
```

Parámetros

value

[in] Valor del dividiendo.

value2

[in] Valor del divisor.

Valor devuelto

La función MathMod calcula el resto real f de val / y de tal manera que val = i * y + f , donde i es un número entero, f tiene el mismo signo que val, y el valor absoluto de f es menos que el valor absoluto de y.

Nota

En vez de la función MathMod() se puede usar la función fmod().

Ejemplo:

```
#property script_show_inputs

//--- input parameters
input double   InpDividentValue  =  10;   // Dividend value
input double   InpDivisorValue   =  3;    // Divisor value

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el resto real de la división de los números introducidos en los paráme
   double res=MathMod(InpDividentValue,InpDivisorValue);
//--- imprimimos el resultado en el diario de registro
   PrintFormat("Real remainder when dividing %.2f by %.2f = %.2f",InpDividentValue,Inp
  }
```

© 2000-2025, MetaQuotes Ltd.

MathPow

Eleva una base a una potencia indicada.

```
double  MathPow(
   double  base,          // base
   double  exponent       // exponente
   );
```

Parámetros

base

[in] Base.

exponent

[in] Valor de potenciación.

Valor devuelto

Valor de la base elevada a la potencia indicada.

Nota

En vez de la función MathPow() se puede usar la función pow().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#property script_show_inputs

#include <Graphics\Graphic.mqh>

//--- input parameters
input double   InpExponentValue  =  2;     // exponent value

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos 11 valores de 0 a 10 con paso 1
   vector X(11,VectorArange);
   Print("vector X = \n",X);
//--- calculamos cada valor del vector X elevado al grado InpExponentValue
   X=MathPow(X,InpExponentValue);
   Print("MathPow(X,",(string)InpExponentValue,") = \n",X);
```

© 2000-2025, MetaQuotes Ltd.

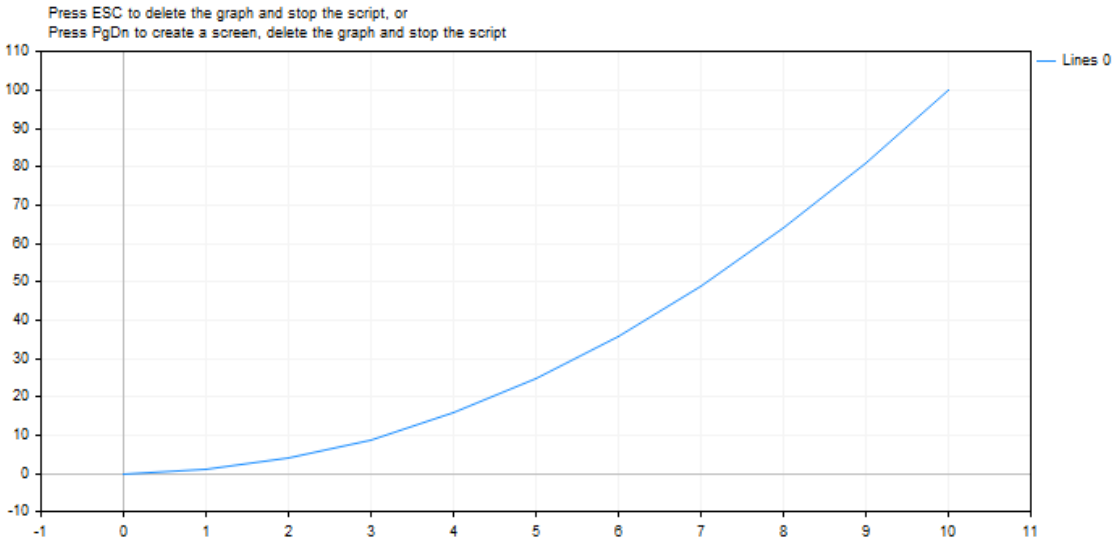
```
//--- transferimos los valores calculados de vector a array
   double y_array[];
   X.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
   /*
  resultado:
   vector X =
   [0,1,2,3,4,5,6,7,8,9,10]
   MathPow(X,2.0) =
   [0,1,4,9,16,25,36,49,64,81,100]
   */
  }
//+------------------------------------------------------------------+
//| Rellena el vector con el valor value con el incremento step      |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:

© 2000-2025, MetaQuotes Ltd.



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathRand

Devuelve un número pseudoaleatorio en el rango de 0 a 32767.

```
int  MathRand();
```

Valor devuelto

Número entero en el rango de 0 a 32767.

Nota

Antes de la primera llamada a la función hay que usar la función MathSrand, con el fin de poner el generador pseudoaleatorio de números en su posición inicial.

Nota

En vez de la función MathRand() se puede usar la función rand().

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- en cada inicio establecemos un nuevo estado inicial para generar una serie de nu
   MathSrand(GetTickCount());
//--- en un ciclo, mostramos en el diario de registro los 10 números enteros pseudoale
   for(int i=0; i<10; i++)
     {
      int rand_value=MathRand();
      PrintFormat("Número entero pseudoaleatorio №%d: %u",i+1,rand_value);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

MathRound

Devuelve un valor redondeado hasta el número entero más cercano del valor numérico especificado.

```
double  MathRound(
   double  value      // valor a redondear
   );
```

Parámetros

value

[in] Valor numérico para ser redondeado.

Valor devuelto

Valor redondeado hasta el número entero más cercano.

Nota

En vez de la función MathRound() se puede usar la función round().

Ejemplo:

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos las variables para la transformación
   double value=0;                     // número real para la transformación MathRound
   int    round_value=0;               // aquí obtenemos el resultado
//--- en un ciclo por el número de incrementos decimales de un número real
   for(int i=0; i<VALUES_TOTAL; i++)
     {
      //--- aumentamos el valor del número real,
      //--- obtenemos un valor numérico redondeado al entero más próximo
      //--- y mostramos en el diario de registro los valores de control
      value+=0.1;
      round_value=(int)MathRound(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, round value: %d",value,round_value);
      /*
     resultado:
      value: 0.1, round value: 0
      value: 0.2, round value: 0
      value: 0.3, round value: 0
      value: 0.4, round value: 0
      value: 0.5, round value: 1
      value: 0.6, round value: 1
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, round value: 1
      value: 0.8, round value: 1
      value: 0.9, round value: 1
      value: 1.0, round value: 1
      value: 1.1, round value: 1
      value: 1.2, round value: 1
      value: 1.3, round value: 1
      value: 1.4, round value: 1
      value: 1.5, round value: 2
      value: 1.6, round value: 2
      value: 1.7, round value: 2
      value: 1.8, round value: 2
      value: 1.9, round value: 2
      value: 2.0, round value: 2
      value: 2.1, round value: 2
      value: 2.2, round value: 2
      value: 2.3, round value: 2
      value: 2.4, round value: 2
      value: 2.5, round value: 3
      value: 2.6, round value: 3
      value: 2.7, round value: 3
      value: 2.8, round value: 3
      value: 2.9, round value: 3
      value: 3.0, round value: 3
      value: 3.1, round value: 3
      */
    }
}
```

© 2000-2025, MetaQuotes Ltd.

MathSin

Devuelve el seno de un ángulo especificado.

```
double  MathSin(
   double  value      // argumento en radianes
   );
```

Parámetros

value

[in] Ángulo en radianes.

Valor devuelto

Seno de un ángulo indicado en radianes. Devuelve valor en el rango de -1 a 1.

Nota

En vez de la función MathSin() se puede usar la función sin().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor del seno para cada valor del vector X
   vector Y=MathSin(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
```

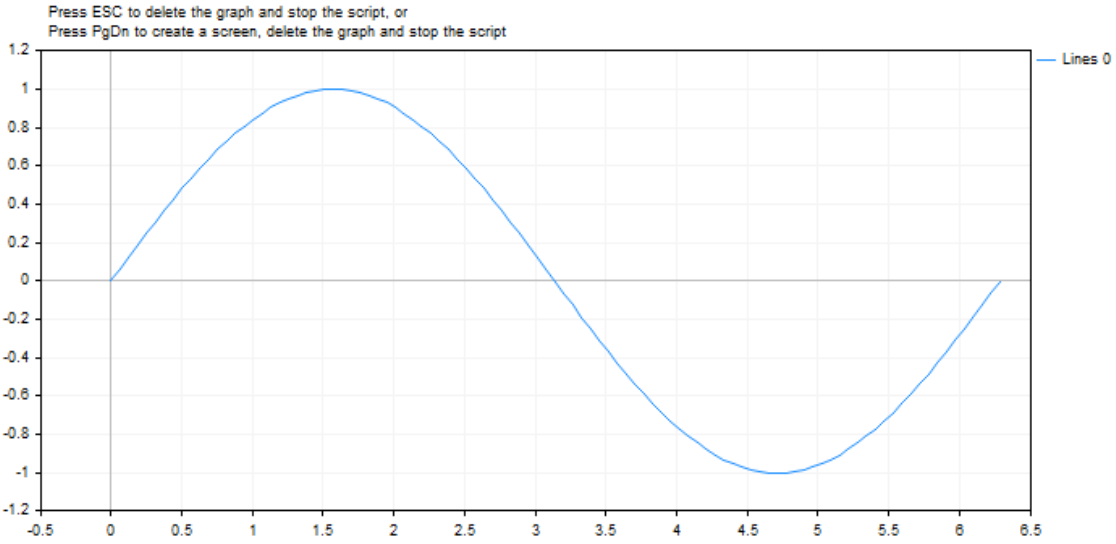
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathSqrt

Devuelve la raíz cuadrada de un número.

```
double  MathSqrt(
   double  value      // número positivo
   );
```

Parámetros

value

[in] Valor numérico positivo.

Valor devuelto

Raíz cuadrada de value. Si value es negativo, MathSqrt devuelve NaN (valor indeterminado).

Nota

En vez de la función MathSqrt() se puede usar la función sqrt().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos 11 valores de 0 a 10 con paso 1
   vector X(11,VectorArange);
   Print("vector X = \n",X);
//--- calculamos la raíz cuadrada para cada valor del vector X
   X=MathSqrt(X);
   Print("MathSqrt(X) = \n",X);

//--- transferimos los valores calculados de vector a array
   double y_array[];
   X.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
   while(!IsStopped())
```

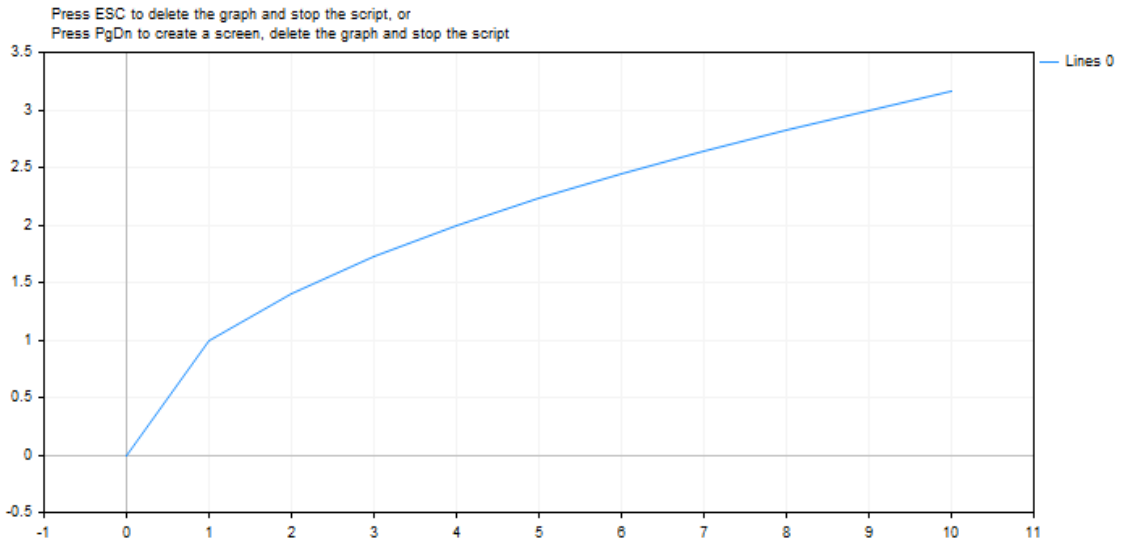
© 2000-2025, MetaQuotes Ltd.

```
      {
       if(StopKeyPressed())
          break;
       Sleep(16);
      }

//--- limpiamos
   ExtGraph.Destroy();
   /*
  resultado:
   vector X =
   [0,1,2,3,4,5,6,7,8,9,10]
   MathSqrt(X) =
   [0,1,1.414213562373095,1.732050807568877,2,2.23606797749979,2.449489742783178,2.645
   */
  }
//+------------------------------------------------------------------+
//| Rellena el vector con el valor value con el incremento step      |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



© 2000-2025, MetaQuotes Ltd.

Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathSrand

Establece el estado inicial para generar una serie de números enteros pseudoaleatorios.

```
void  MathSrand(
   int  seed      // número de inicialización
   );
```

Parámetros

seed

[in] Número inicial para una fila de números aleatorios.

Valor devuelto

No hay valor devuelto.

Nota

La función MathRand() sirve para generar una secuencia de números pseudoaleatorios. La llamada de MathSrand() con un cierto número inicializador permite recibir siempre la misma secuencia de números pseudoaleatorios.

Para garantizar la recepción de una secuencia irrepetible, utilice la llamada de MathSrand(GetTickCount()), ya que el valor GetTickCount() se aumenta desde el arranque del sistema operativo y no se repite durante 49 días hasta que se llene el contador incorporado de milisegundos. El uso de MathSrand(TimeCurrent()) no vale porque la función TimeCurrent() devuelve la hora de llegada del último tick que puede ser intacto durante bastante tiempo, por ejemplo durante el fin de semana.

La inicialización del generador de números pseudoaleatorios con el uso de MathSrand() para los indicadores y EAs es mejor hacer dentro del manejador OnInit(). Esto permite evitar los posteriores numerosos arranques del generador en OnTick() y OnCalculate().

En vez de la función MathSrand() se puede usar la función srand().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#property description "El indicador demuestra el teorema del límite central que indica
#property description "La suma de un número suficientemente grande de variables aleato
#property description "que tienen aproximadamente las mismas dimensiones (ninguno de l
#property description "sin ejercer ninguna contribución determinante en la suma), «se

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- propiedades de construcción gráfica
#property indicator_label1  "Label"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRoyalBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  5
//--- variable input
input int      sample_number=10;
//--- búfer indicador para dibujar la distribución
double         LabelBuffer[];
//--- contador de ticks
double         ticks_counter;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- enlace del array con el búfer de indicador
   SetIndexBuffer(0,LabelBuffer,INDICATOR_DATA);
//--- damos la vuelta al búfer de indicador desde el presente hasta el pasado
   ArraySetAsSeries(LabelBuffer,true);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- inicializamos el contador de ticks
   ticks_counter=0;
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- con el contador cero ponemos el búfer de indicador a cero
   if(ticks_counter==0) ArrayInitialize(LabelBuffer,0);
//--- aumentamos el contador
   ticks_counter++;
//--- hace falta poner a cero periódicamente el contador de ticks para reavivar la dis
   if(ticks_counter>100)
     {
      Print("Tras poner a cero los valores del indicador, volvemos a rellenar las celd
      ticks_counter=0;
     }
//--- obtenemos la muestra de valores aleatorios como la suma de tres números de 0 a 1
   for(int i=0;i<sample_number;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- cálculo del índice de la celda en la que se coloca el número aleatorio cor
      int rand_index=0;
      //--- obtenemos tres números aleatorios de 0 a 7
      for(int k=0;k<3;k++)
        {
         //--- el resto de la división por 7 devolverá un valor de 0 a 6
         rand_index+=MathRand()%7;
        }
      //--- aumentamos a uno el valor en la celda con el número rand_index
      LabelBuffer[rand_index]++;
     }
//--- salimos del manejador OnCalculate()
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathTan

Devuelve la tangete de un número.

```
double  MathTan(
   double  rad      // argumento en radianes
   );
```

Parámetros

rad

[in] Ángulo en radianes.

Valor devuelto

Tangete del número rad. Si rad es más de o igual a 263 o menos de o igual a -263, ocurre la perdida del valor y la función devuelve un número indeterminado.

Nota

En vez de la función MathTan() se puede usar la función tan().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=-263;
//--- obtenemos 101 valores de -263 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor de la tangente para cada valor del vector X
   vector Y=MathTan(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

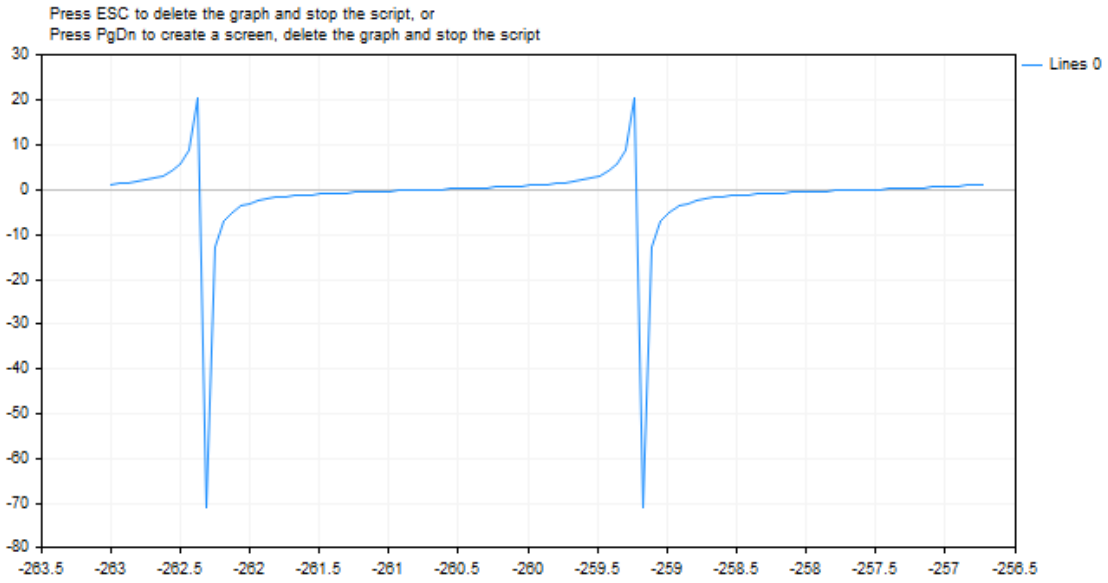
© 2000-2025, MetaQuotes Ltd.

```
//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRIT
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathIsValidNumber

Verifica la correctitud de un número real.

```
bool  MathIsValidNumber(
   double  number      // número a comprobar
   );
```

Parámetros

number

[in] Número a comprobar.

Valor devuelto

Devuelve true, si el valor que se comprueba es un número real aceptable. Si el valor que se comprueba es más o menos infinito, o se trata de "no número" (NaN - not a number), la función devuelve false.

Ejemplo:

```
   double abnormal=MathArcsin(2.0);
   if(!MathIsValidNumber(abnormal)) Print("Atención! MathArcsin(2.0) =",abnormal);
```

Véase también

Tipos reales (double, float)

© 2000-2025, MetaQuotes Ltd.

MathExpm1

Devuelve el valor de la expresión MathExp(x)-1.

```
double  MathExpm1(
   double  value      // exponente para el número e
   );
```

Parámetros

value

[in] Número que define el exponente.

Valor devuelto

Número del tipo double. En caso de sobrecarga, la función devuelve INF (infinito), en caso de pérdida en el grado de precisión, MathExpm1 devuelve 0.

Nota

Con valores *x* cercanos a 0, la función MathExpm1(x) da valores mucho más precisos que MathExp(x)-1.

En lugar de la función MathExpm1(), se puede usar la función expm1().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos 9 valores de 0 a 8 con paso 1
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- calculamos el valor ("e" (número de Euler) en el grado de cada valor del vector
   X=MathExpm1(X);
   Print("MathExpm1(X) = \n",X);

//--- transferimos los valores calculados de vector a array
   double y_array[];
   X.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
```

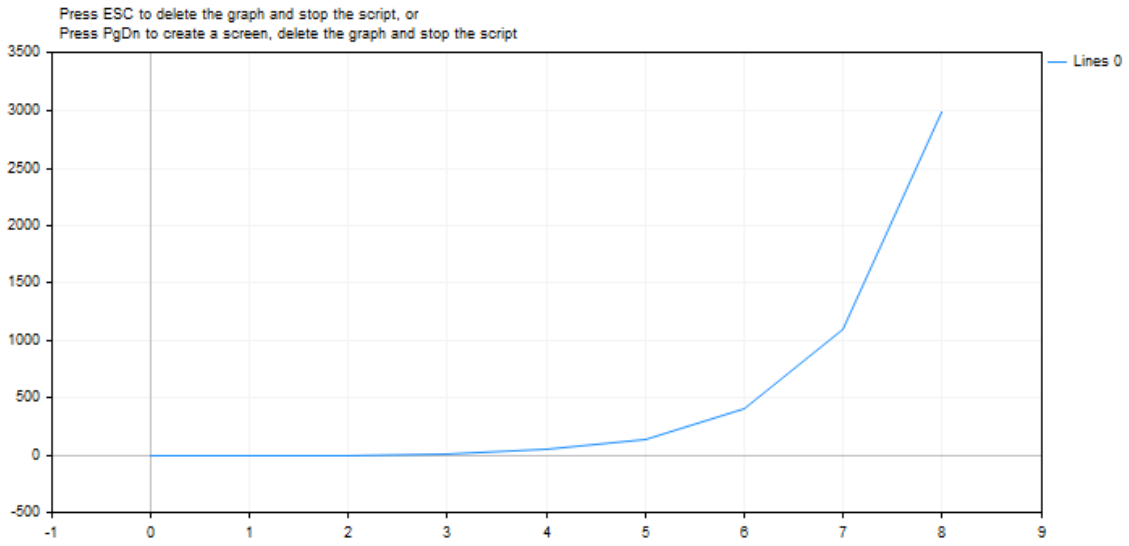
© 2000-2025, MetaQuotes Ltd.

```
   CurvePlot(y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Rellena el vector con el valor value con el incremento step      |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRIT
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathLog1p

Devuelve el valor de la expresión MathLog(1+x).

```
double  MathLog1p(
   double  value      // número para coger el logaritmo
   );
```

Parámetros

value

[in] Valor cuyo logaritmo debe ser calculado.

Valor devuelto

Logaritmo natural del valor (value+1) en caso de éxito. Si value < -1, la función devuelve NaN (valor indeterminado). Si value es igual a -1, la función devuelve INF (infinito).

Nota

Con valores x cercanos a 0, la función MathLog1p(x) da valores mucho más precisos que MathLog(1+x)

En lugar de la función MathLog1p(), se puede usar la función log1p().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos 9 valores de 0 a 8 con paso 1
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- para cada valor del vector calculamos el logaritmo para la expresión (1 + valor
   X=MathLog1p(X);
   Print("MathLog1p(X) = \n",X);

//--- transferimos los valores calculados de vector a array
   double y_array[];
   X.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
```

© 2000-2025, MetaQuotes Ltd.

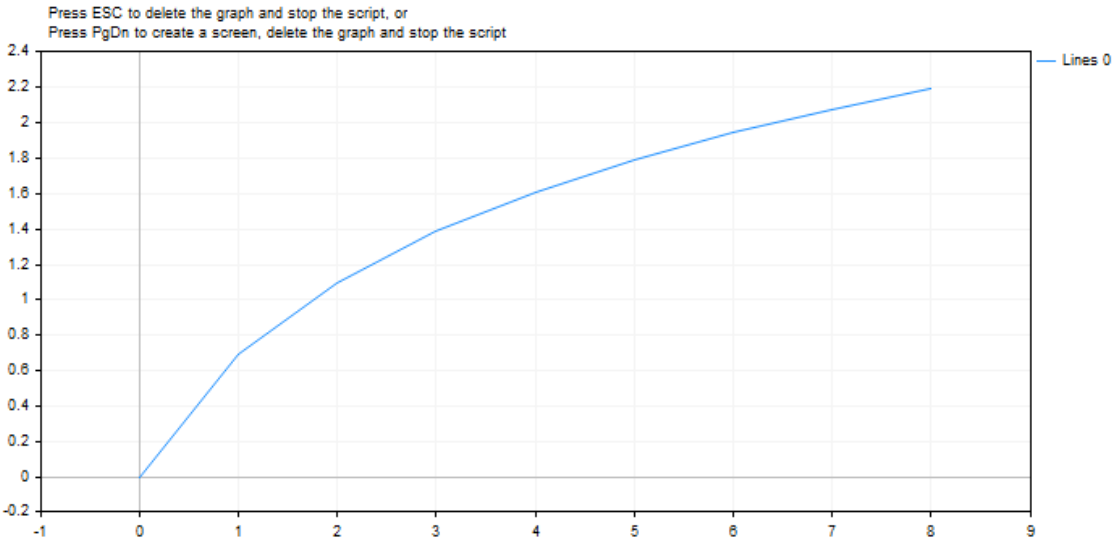
```
   CurvePlot(y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
   /*
  resultado:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog1p(X) =
   [0,0.6931471805599453,1.09861228866811,1.386294361119891,1.6094379124341,1.7917594
   */
  }
//+------------------------------------------------------------------+
//| Rellena el vector con el valor value con el incremento step      |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:

© 2000-2025, MetaQuotes Ltd.



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathArccosh

Devuelve el valor del arcocoseno hiperbólico.

```
double  MathArccosh(
   double  value     // 1 <= value < ∞
   );
```

Parámetros

val

[in] Valor value, cuyo arcocoseno hiperbólico debe ser calculado.

Valor devuelto

Arcocoseno hiperbólico del número. Si value es menor a +1, la función retorna NaN (valor indeterminado).

Nota

En lugar de la función MathArccosh(), se puede usar la función acosh().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/10);
   delta[0]=1;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor del arcocoseno hiperbólico para cada valor del vector X
   vector Y=MathArccosh(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

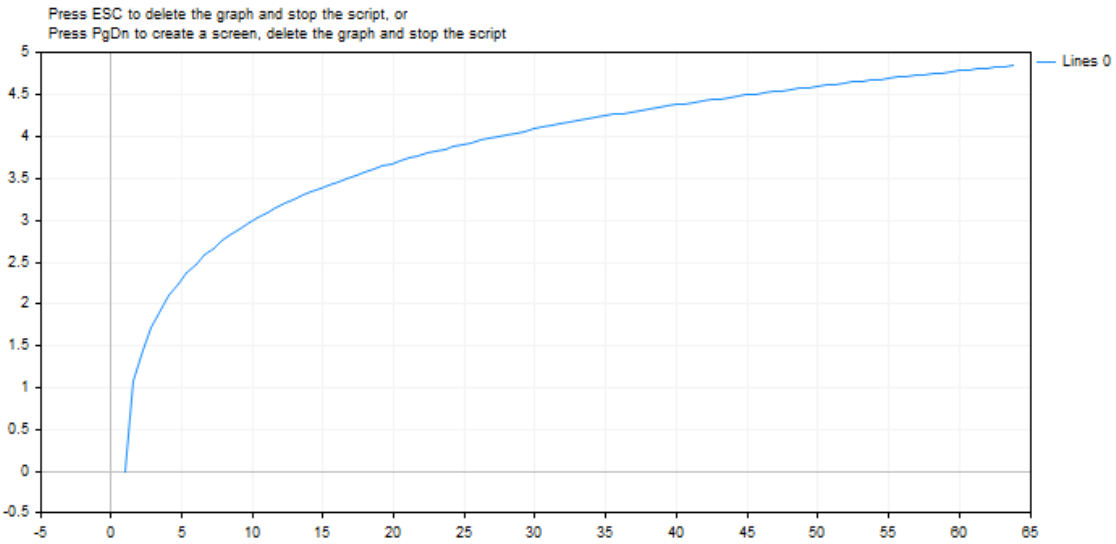
© 2000-2025, MetaQuotes Ltd.

```
//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathArcsinh

Devuelve el valor del arcoseno hiperbólico.

```
double  MathArcsinh(
   double  value      // -∞ < value < +∞
   );
```

Parámetros

value

[in] Valor value, cuyo arcoseno hiperbólico debe ser calculado.

Valor devuelto

Arcoseno hiperbólico del número.

Nota

En lugar de la función MathArcsinh(), se puede usar la función asinh().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/10);
   delta[0]=-31;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor del arcoseno hiperbólico para cada valor del vector X
   vector Y=MathArcsinh(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
```

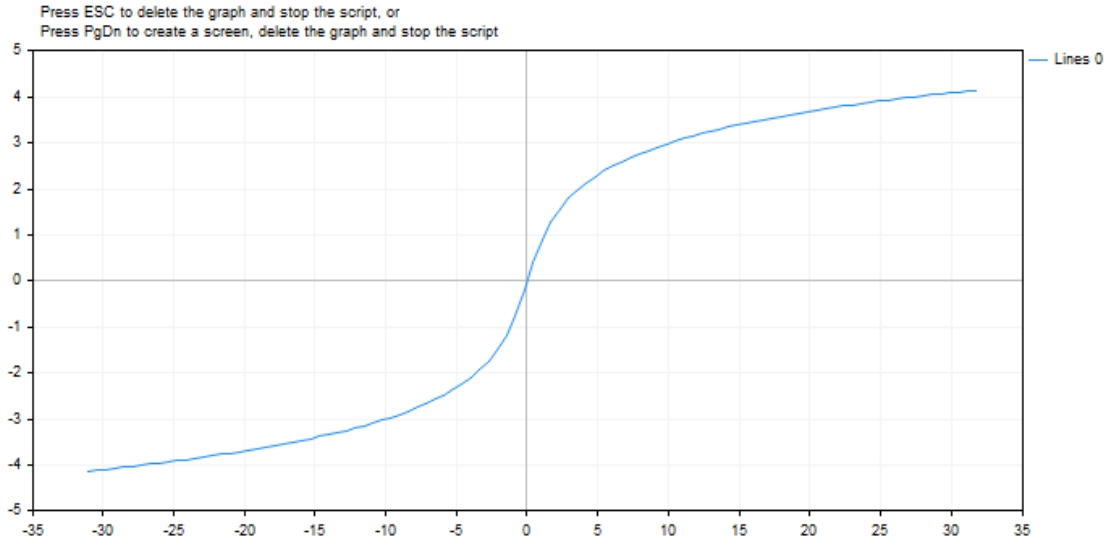
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathArctanh

Devuelve el valor de la arcotangente hiperbólica.

```
double  MathArctanh(
   double  value      // valor en el rango -1 < value < 1
   );
```

Parámetros

value

[in] Número en el rango -1 < value < 1, que representa la tangente.

Valor devuelto

Arcotangente hiperbólica del número.

Nota

En lugar de la función MathArctanh(), se puede usar la función atanh().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(34,2*M_PI/100);
   delta[0]=-1;
//--- obtenemos 34 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor de la arcotangente hiperbólica para cada valor del vector X
   vector Y=MathArctanh(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
```

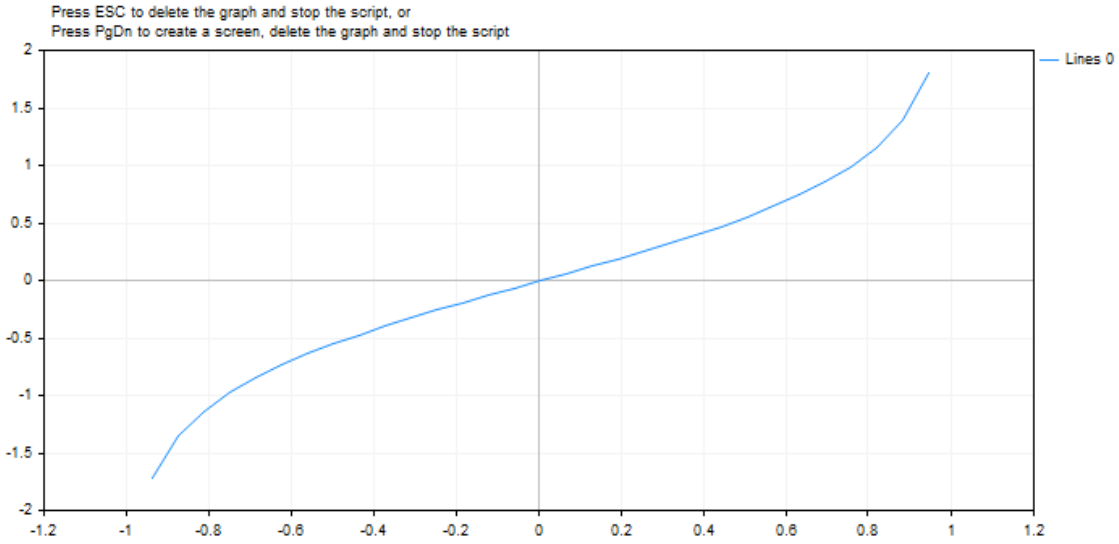
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathCosh

Devuelve el coseno hiperbólico del número.

```
double  MathCosh(
   double  value      // número
   );
```

Parámetros

value

[in] Valor.

Valor devuelto

Coseno hiperbólico del número, valor en el rango de +1 hasta más infinito.

Nota

En lugar de la función MathCosh(), se puede usar la función cosh().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor del coseno hiperbólico para cada valor del vector X
   vector Y=MathCosh(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
```

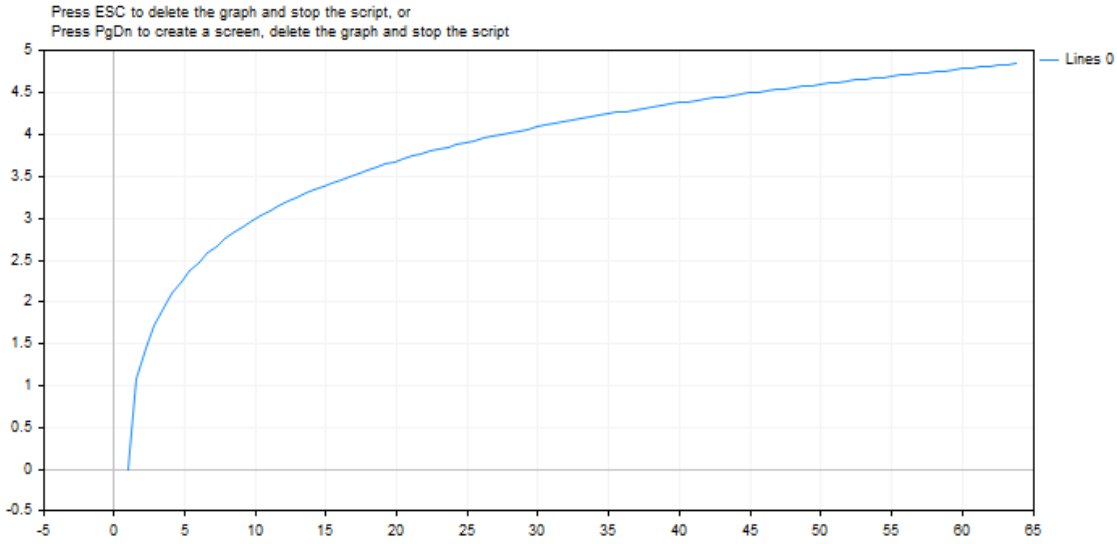
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathSinh

Devuelve el seno hiperbólico del número.

```
double  MathSinh(
   double  value      // número
   );
```

Parámetros

value

[in] Valor.

Valor devuelto

Seno hiperbólico del número.

Nota

En lugar de la función MathSinh(), se puede usar la función sinh().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor del seno hiperbólico para cada valor del vector X
   vector Y=MathSinh(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
```

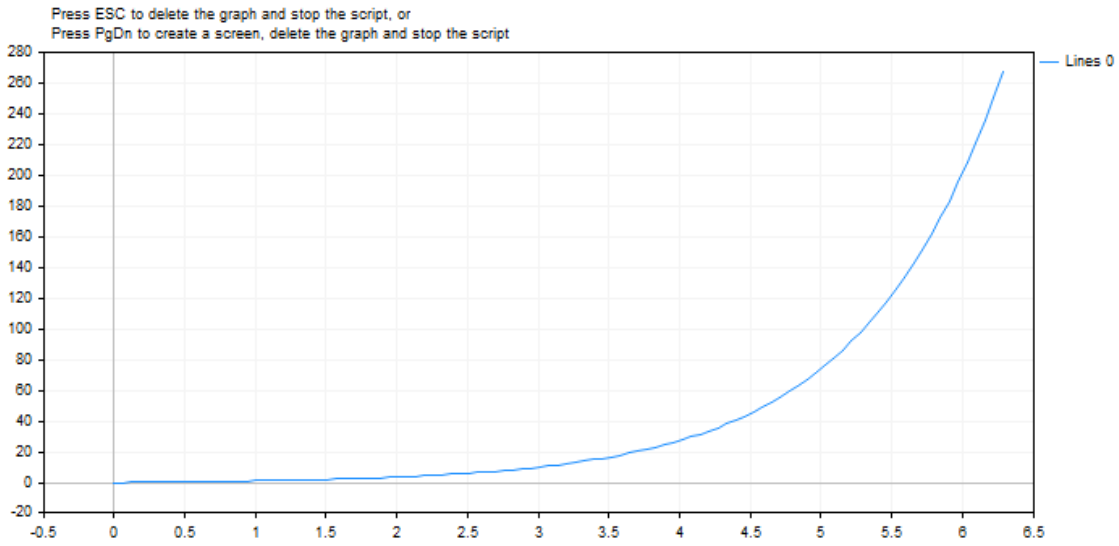
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathTanh

Devuelve la tangente hiperbólica del número.

```
double  MathTanh(
   double  value      // número
   );
```

Parámetros

value

[in] Valor.

Valor devuelto

Tangente hiperbólica del número, valor en el rango de -1 a +1.

Nota

En lugar de la función MathTanh(), se puede usar la función tanh().

Ejemplo:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- obtenemos 101 valores de -1 a 2 pi con paso delta
   vector X=delta.CumSum();
//--- calculamos el valor de la tangente hiperbólica para cada valor del vector X
   vector Y=MathTanh(X);

//--- transferimos los valores calculados de vectores a arrays
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- dibujamos el gráfico de los valores calculados del vector
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- esperamos la pulsación de Escape o PgDn para eliminar el gráfico (crear una capt
```

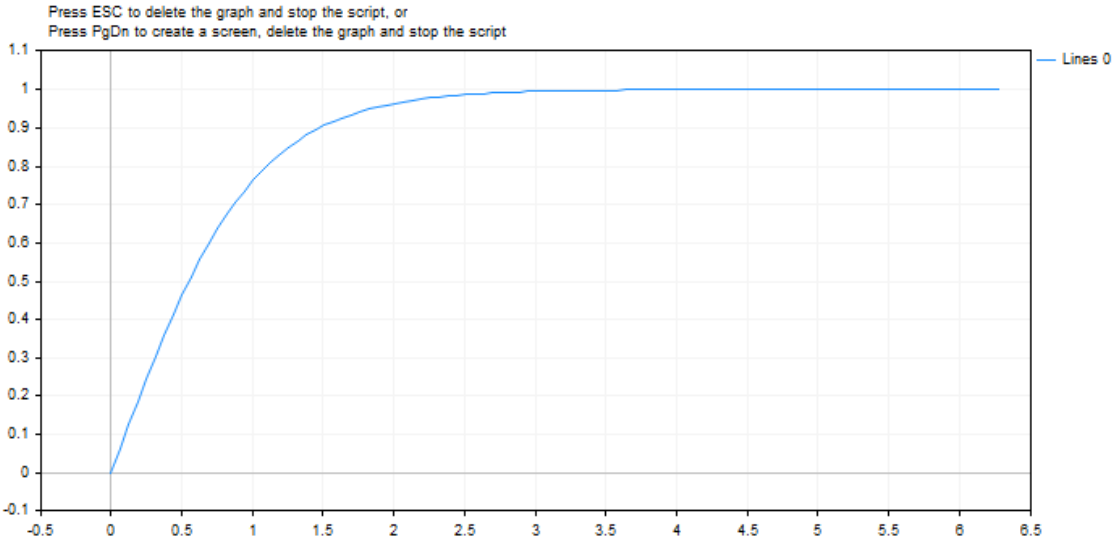
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- limpiamos
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Al pulsar ESC, retorna true                                      |
//| Al pulsar PgDn hace una captura del gráfico y retorna true       |
//| Si no, retorna false                                             |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- si ESC está pulsado, retornamos true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- si PgDn está pulsado y se realiza correctamente la captura de pantalla, retornar
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- retornamos false
   return(false);
  }
//+------------------------------------------------------------------+
//| Crea un objeto de gráfico y dibuja una curva                     |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Crea una captura de pantalla y guarda la imagen en un archivo    |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Resultado:



Vea también

Tipos reales (double, float), Estadística, Gráficos científicos, Estado del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

MathSwap

Cambia el orden de los bytes en un valor del tipo ushort.

```
ushort  MathSwap(
   ushort  value      // valor
   );
```

Parámetros

value

[in] Valor para el cambio de orden de los bytes.

Valor retornado

Valor ushort con orden de bytes inverso.

MathSwap

Cambia el orden de los bytes en un valor del tipo uint.

```
uint  MathSwap(
   uint   value      // valor
   );
```

Parámetros

value

[in] Valor para el cambio de orden de los bytes.

Valor retornado

Valor uint con orden de bytes inverso.

MathSwap

Cambia el orden de los bytes en un valor del tipo ulong.

```
ulong  MathSwap(
   ulong  value      // valor
   );
```

Parámetros

value

[in] Valor para el cambio de orden de los bytes.

Valor retornado

Valor ulong con orden de bytes inverso.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#property script_show_inputs

input ulong  InpLongValue  =  1;     // Enter any ulong value here
input uint   InpIntValue   =  2;     // Enter any uint value here
input ushort InpShortValue =  3;     // Enter any ushort value here

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {

//--- imprimimos en el diario de registro los valores introducidos y convertidos por M
   Print(ValueDescription(InpLongValue));
   Print(ValueDescription(InpIntValue));
   Print(ValueDescription(InpShortValue));
   /*
  resultado:
   ulong value: 1
   ulong value: 72057594037927936 using MathSwap()
   binary ulong value: 0000000000000000000000000000000000000000000000000000000000000000
   binary ulong value: 0000000100000000000000000000000000000000000000000000000000000000

   uint value: 2
   uint value: 33554432 using MathSwap()
   binary uint value: 00000000000000000000000000000010
   binary uint value: 00000010000000000000000000000000 using MathSwap()

   ushort value: 3
   ushort value: 768 using MathSwap()
   binary ushort value: 0000000000000011
   binary ushort value: 0000001100000000 using MathSwap()
   */
  }
//+------------------------------------------------------------------+
//| Devuelve la descripción textual de los valores de la variable    |
//+------------------------------------------------------------------+
template <typename T>
string ValueDescription(T x)
  {
   int    num_bits   = sizeof(T)*8;
   string type_name  = typename(T);
   string bin_x      = NumberToBinaryString(x);
   string bin_swap_x = NumberToBinaryString(MathSwap(x));
   return(StringFormat("%s value: %lld\n%s value: %lld using MathSwap()\nbinary %s val
          type_name, x, type_name, MathSwap(x), type_name, num_bits, bin_x, type_name,
  }
//+------------------------------------------------------------------+
//| Devuelve la representación binaria del número como cadena        |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
template <typename T>
string NumberToBinaryString(T x)
  {
   string res  = "";
   int    i    = -1;
   uchar  size = sizeof(T)*8-1;
   ulong  mask = (ulong)1<<size;
   while(!((x<<++i)& mask));
   for(; i <=size; i++)
      res += !((x<<i)& mask) ? "0" : "1";
   return res;
  }
```

Ver también

Funciones de red, SocketRead, SocketSend, SocketTlsRead, SocketTlsReadAvailable, SocketTlsSend

© 2000-2025, MetaQuotes Ltd.

Funciones literales

Es un grupo de funciones que operan con los datos del tipo string.

Función	Acción
StringAdd	Añade una subcadena especificada a una cadena en el lugar
StringBufferLen	Devuelve el tamaño del buffer distribuido para una cadena
StringCompare	Compara dos cadenas de caracteres, y devuelve 1 si la primera cadena es más grande que la segunda, devuelve 0 si las cadenas son iguales, y -1 (menos 1) si la primera es menor que la segunda.
StringConcatenate	Forma una cadena con los parámetros pasados
StringFill	Rellena una cadena especificada con símbolos especificados en el lugar
StringFind	Busca una subcadena en una cadena
StringGetCharacter	Devuelve el valor de un símbolo que se encuentra en la posición especificada de una cadena
StringInit	Inicializa una cadena con símbolos especificados y proporciona la longitud especificada de una cadena
StringLen	Devuelve el número de símbolos de una cadena
StringSetLength	Establece para una cadena la longitud indicada en símbolos
StringReplace	Reemplaza todas las subcadenas encontradas en una cadena de caracteres por una secuencia de símbolos.
StringReserve	Reserva en la memoria para la cadena un búfer del tamaño indicado
StringSetCharacter	Devuelve copia de una cadena con el valor del símbolo modificado en una posición especificada
StringSplit	Obtiene las subcadenas por un separador establecido desde la cadena especificada y devuelve el número de subcadenas obtenidas
StringSubstr	Extrae una subcadena de una cadena de texto que se empieza desde una posición especificada
StringToLower	Transforma todos los símbolos de una cadena indicada en minúsculas en el lugar
StringToUpper	Transforma todos los símbolos de una cadena indicada en mayúsculas en el lugar
StringTrimLeft	Borra los símbolos de salto de línea, espacios y símbolos de tabulación en la parte izquierda de la cadena
StringTrimRight	Borra los símbolos de salto de línea, espacios y símbolos de tabulación en la parte derecha de la cadena

© 2000-2025, MetaQuotes Ltd.

StringAdd

La función añade una subcadena especificada al final de una cadena en el lugar.

```
bool  StringAdd(
   string&  string_var,        // cadena a la que añadimos
   string   add_substring      // cadena que se añade
   );
```

Parámetros

string_var

[in][out] Cadena que va a ser completada con otra.

add_substring

[in] Cadena que va a ser añadida al final de la cadena fuente.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false. Para obtener el código de error hay que llamar a la función GetLastError().

Ejemplo:

```
void OnStart()
  {
   long length=1000000;
   string a="a",b="b",c;
//--- primer método
   uint start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);

//--- segundo método
   start=GetTickCount();
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   stop=GetTickCount();
   Print("time for 'StringAdd(a,b)' = ",(stop-start)," milliseconds, i = ",i);

//--- tercer método
   start=GetTickCount();
   a="a"; // volvemos a inicializar la variable a
   for(i=0;i<length;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       StringConcatenate(c,a,b);
      }
   stop=GetTickCount();
   Print("time for 'StringConcatenate(c,a,b)' = ",(stop-start),"milliseconds, i = ",i)
  }
```

Véase también

StringConcatenate, StringSplit, StringSubstr

© 2000-2025, MetaQuotes Ltd.

StringBufferLen

Devuelve el tamaño del buffer distribuido para una cadena.

```
int  StringBufferLen(
   string  string_var      // cadena
   )
```

Parámetros

string_var

[in] Cadena.

Valor devuelto

El valor 0 significa que esta cadena es una cadena constante y no se puede cambiar el contenido del buffer. -1 significa que la cadena pertenece al terminal de cliente y el cambio del contenido del buffer puede suponer unos resultados indeterminados.

Ejemplo:

```
void OnStart()
  {
   long length=1000;
   string a="a",b="b";
//---
   long i;
   Print("before: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   Print("after: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
  }
```

Véase también

StringAdd, StringInit, StringLen, StringFill

© 2000-2025, MetaQuotes Ltd.

StringCompare

Esta función compara dos cadenas de caracteres y devuelve el resultado de la comparación en forma de un número entero.

```
int  StringCompare(
   const string&  string1,                // la primera cadena a comparar
   const string&  string2,                // la segunda cadena a comparar
   bool           case_sensitive=true     // modo de sensibilidad a mayúsculas
   );
```

Parámetros

string1

[in] La primera cadena.

string2

[in] La segunda cadena.

case_sensitive=true

[in] Selección del modo de sensibilidad a mayúsculas. Si es true, entonces "A">"a". Si es false, entonces "A"="a". Por defecto, el valor de este parámetro es igual a true.

Valor devuelto

- -1 (menos uno), si string1<string2
- 0 (cero), si string1=string2
- 1 (uno), si string1>string2

Nota

Las cadenas se comparan carácter por carácter. Los caracteres se comparan en orden alfabético de acuerdo con la página de código actual.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- ¿Qué es más grande, una manzana o una casa?
   string s1="Apple";
   string s2="home";

//--- modo de sensibilidad a mayúsculas activado
   int result1=StringCompare(s1,s2);
   if(result1>0) PrintFormat("Comparación sensible a mayúsculas: %s > %s",s1,s2);
   else
     {
      if(result1<0)PrintFormat("Comparación sensible a mayúsculas: %s < %s",s1,s2);
      else PrintFormat("Comparación sensible a mayúsculas: %s = %s",s1,s2);
     }

//--- modo de sensibilidad a mayúsculas desactivado
   int result2=StringCompare(s1,s2,false);
   if(result2>0) PrintFormat("Comparación no sensible a mayúsculas: %s > %s",s1,s2);
   else
     {
      if(result2<0)PrintFormat("Comparación no sensible a mayúsculas: %s < %s",s1,s2);
      else PrintFormat("Comparación no sensible a mayúsculas: %s = %s",s1,s2);
     }
/* Resultado
     Comparación sensible a mayúsculas: Apple < home
     Comparación no sensible a mayúsculas: Apple < home
*/
  }
```

Véase también

Tipo string, CharToString(), ShortToString(), StringToCharArray(), StringToShortArray(), StringGetCharacter(), Uso de página de código

© 2000-2025, MetaQuotes Ltd.

StringConcatenate

Forma una cadena con los parámetros pasados y devuelve el tamaño de la cadena formada. Los parámetros pueden ser de cualquier tipo. El número de parámetros no puede ser menos de 2 y más de 64.

```
int  StringConcatenate(
   string&  string_var,    // cadena para formar
   void argument1          // primer parámetro de cualquier tipo simple
   void argument2          // segundo parámetro de cualquier tipo simple
   ...                     // siguiente parámetro de cualquier tipo simple
   );
```

Parámetros

string_var

[out] Cadena que va a ser formada como resultado de concatenación.

argumentN

[in] Cualquieres valores separados por comas. De 2 a 63 parámetros de cualquier tipo simple.

Valor devuelto

Devuelve la longitud de la cadena formada por medio de la concatenación de parámetros transformados en el tipo string. Los parámetros se transforman en las cadenas según las mismas reglas que en las funciones Print() y Comment().

Ejemplo:

```
void OnStart()
  {
//--- declaramos y definimos las variables que intervienen en la concatenación
   string text="";
   string text1="This script shows how the StringConcatenate() function works.\n";
   string text2="This is the second line, at the end of which there is a line break co
   string text3="This is line number ";
   int    num3=3;
   string text31=", the number of which is entered into the function as a separate pa
   string textN="\n";
   string text4="This is line number 4, preceded by a separate parameter with a line b
   int    length=StringConcatenate(text, text1, text2, text3, num3, text31, textN, tex
   Print(text, "\nLength of the resulting string = ", length);

  /*
  Resultado
   This script shows how the StringConcatenate() function works.
   This is the second line, at the end of which there is a line break control code.
   This is line number 3, the number of which is entered into the function as a separa
   This is line number 4, preceded by a separate parameter with a line break code.
   Line 5 includes a real number: 0.12345
   Length of the resulting string = 358
```

© 2000-2025, MetaQuotes Ltd.

```
   */
   }
```

Véase también

StringAdd, StringSplit, StringSubstr

© 2000-2025, MetaQuotes Ltd.

StringFill

Llena una cadena especificada con símbolos especificados en el lugar.

```
bool  StringFill(
   string&   string_var,       // cadena para llenar
   ushort    character         // símbolos que van a llenar la cadena
   );
```

Parámetros

string_var

[in][out] Cadena que va a ser llenada con símbolos especificados.

character

[in] Símbolo con el que va a ser llenada la cadena.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false. Para obtener el código de error hay que llamar a la función GetLastError().

Nota

Llenar una cadena en el lugar significa que los símbolos se insertan directamente en la cadena sin operaciones intermedias de creación de unas nuevas cadenas y sin copiado. Esto permite ahorrar el tiempo de trabajo con una cadena en dicha función.

Ejemplo:

```
void OnStart()
  {
   string str;
   StringInit(str,20,'_');
   Print("str = ",str);
   StringFill(str,0);
   Print("str = ",str,": StringBufferLen(str) = ", StringBufferLen(str));
  }
// Resultado
//   str = ____________________
//   str =  : StringBufferLen(str) = 20
//
```

Véase también

StringBufferLen, StringLen, StringInit

© 2000-2025, MetaQuotes Ltd.

StringFind

Busca una subcadena en una cadena.

```
int  StringFind(
   string  string_value,        // cadena en la que buscamos
   string  match_substring,     // lo que buscamos
   int     start_pos=0          // punto de partida de la búsqueda
   );
```

Parámetros

string_value

[in] Cadena en la que se realiza la búsqueda.

match_substring

[in] Subcadena buscada.

start_pos=0

[in] Posición en la cadena desde la cual debe empezarse la búsqueda.

Valor devuelto

Devuelve el número de posición en la cadena desde la cual se empieza la subcadena buscada, o devuelve -1 si la subcadena no ha sido encontrada.

Ejemplo:

```
#define   RESERVE    100

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos la divisa básica del instrumento y la divisa del beneficio
   string symbol_currency_base  =SymbolInfoString(Symbol(), SYMBOL_CURRENCY_BASE);
   string symbol_currency_profit=SymbolInfoString(Symbol(), SYMBOL_CURRENCY_PROFIT);
   PrintFormat("Symbol Currency Base: %s\nSymbol Currency Profit: %s", symbol_currency

//--- en un ciclo a través de todos los símbolos disponibles en el servidor
   int total=SymbolsTotal(false), pos=-1;
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el nombre del símbolo siguiente
      string name=SymbolName(i, false);

      //--- buscamos en el nombre del símbolo una subcadena con el nombre de la divisa
      //--- si se encuentra la subcadena, imprimimos el nombre del instrumento, su ín
      pos = StringFind(name, symbol_currency_base);
      if(pos >= 0)
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("The '%s' symbol at index %u in the list contains the '%s' currer

      //--- buscamos en el nombre del símbolo una subcadena con el nombre de la divisa
      //--- si se encuentra la subcadena, imprimimos el nombre del instrumento, su índ
      pos = StringFind(name, symbol_currency_profit);
      if(pos >= 0)
         PrintFormat("The '%s' symbol at index %u in the list contains the '%s' currer
     }

  /*
Resultado
  StringFind (EURUSD,D1)    Symbol Currency Base: EUR
  StringFind (EURUSD,D1)    Symbol Currency Profit: USD
  The 'EURUSD' symbol at index 0 in the list contains the 'EUR' currency. Substring p
  The 'EURUSD' symbol at index 0 in the list contains the 'USD' currency. Substring p
  The 'GBPUSD' symbol at index 1 in the list contains the 'USD' currency. Substring p
  The 'USDCHF' symbol at index 2 in the list contains the 'USD' currency. Substring p
  The 'USDJPY' symbol at index 3 in the list contains the 'USD' currency. Substring p
  The 'USDCNH' symbol at index 4 in the list contains the 'USD' currency. Substring p
  The 'USDRUB' symbol at index 5 in the list contains the 'USD' currency. Substring p
  The 'AUDUSD' symbol at index 6 in the list contains the 'USD' currency. Substring p
  The 'NZDUSD' symbol at index 7 in the list contains the 'USD' currency. Substring p
  The 'USDCAD' symbol at index 8 in the list contains the 'USD' currency. Substring p
  The 'USDSEK' symbol at index 9 in the list contains the 'USD' currency. Substring p
  The 'USDHKD' symbol at index 10 in the list contains the 'USD' currency. Substring
  The 'USDSGD' symbol at index 11 in the list contains the 'USD' currency. Substring
  The 'USDNOK' symbol at index 12 in the list contains the 'USD' currency. Substring
  The 'USDDKK' symbol at index 13 in the list contains the 'USD' currency. Substring
  The 'USDTRY' symbol at index 14 in the list contains the 'USD' currency. Substring
  The 'USDZAR' symbol at index 15 in the list contains the 'USD' currency. Substring
  The 'USDCZK' symbol at index 16 in the list contains the 'USD' currency. Substring
  The 'USDHUF' symbol at index 17 in the list contains the 'USD' currency. Substring
  The 'USDPLN' symbol at index 18 in the list contains the 'USD' currency. Substring
  The 'USDRUR' symbol at index 19 in the list contains the 'USD' currency. Substring
  The 'EURAUD' symbol at index 27 in the list contains the 'EUR' currency. Substring
  The 'EURCAD' symbol at index 28 in the list contains the 'EUR' currency. Substring
  The 'EURCHF' symbol at index 29 in the list contains the 'EUR' currency. Substring
  The 'EURCZK' symbol at index 30 in the list contains the 'EUR' currency. Substring
  The 'EURDKK' symbol at index 31 in the list contains the 'EUR' currency. Substring
  The 'EURGBP' symbol at index 32 in the list contains the 'EUR' currency. Substring
  The 'EURHKD' symbol at index 33 in the list contains the 'EUR' currency. Substring
  The 'EURHUF' symbol at index 34 in the list contains the 'EUR' currency. Substring
  The 'EURJPY' symbol at index 35 in the list contains the 'EUR' currency. Substring
  The 'EURNOK' symbol at index 36 in the list contains the 'EUR' currency. Substring
  The 'EURNZD' symbol at index 37 in the list contains the 'EUR' currency. Substring
  The 'EURPLN' symbol at index 38 in the list contains the 'EUR' currency. Substring
  The 'EURRUR' symbol at index 39 in the list contains the 'EUR' currency. Substring
  The 'EURRUB' symbol at index 40 in the list contains the 'EUR' currency. Substring
  The 'EURSEK' symbol at index 41 in the list contains the 'EUR' currency. Substring
```

© 2000-2025, MetaQuotes Ltd.

```
   The 'EURTRY' symbol at index 42 in the list contains the 'EUR' currency. Substring
   The 'EURZAR' symbol at index 43 in the list contains the 'EUR' currency. Substring
   The 'XAUUSD' symbol at index 47 in the list contains the 'USD' currency. Substring
   The 'XAUEUR' symbol at index 48 in the list contains the 'EUR' currency. Substring
   The 'XAGUSD' symbol at index 50 in the list contains the 'USD' currency. Substring
   The 'XAGEUR' symbol at index 51 in the list contains the 'EUR' currency. Substring
   The 'USDCRE' symbol at index 53 in the list contains the 'USD' currency. Substring
   The 'XPDUSD' symbol at index 65 in the list contains the 'USD' currency. Substring
   The 'XPTUSD' symbol at index 66 in the list contains the 'USD' currency. Substring
   The 'USDGEL' symbol at index 67 in the list contains the 'USD' currency. Substring
   The 'USDMXN' symbol at index 68 in the list contains the 'USD' currency. Substring
   The 'EURMXN' symbol at index 69 in the list contains the 'EUR' currency. Substring
   The 'USDCOP' symbol at index 75 in the list contains the 'USD' currency. Substring
   The 'USDARS' symbol at index 76 in the list contains the 'USD' currency. Substring
   The 'USDCLP' symbol at index 77 in the list contains the 'USD' currency. Substring
   The 'EURSGD' symbol at index 89 in the list contains the 'EUR' currency. Substring
   The 'USDILS' symbol at index 95 in the list contains the 'USD' currency. Substring
   The 'USDTHB' symbol at index 122 in the list contains the 'USD' currency. Substring
   The 'USDRMB' symbol at index 123 in the list contains the 'USD' currency. Substring
   The 'EURILS' symbol at index 126 in the list contains the 'EUR' currency. Substring
   The 'EURCNH' symbol at index 137 in the list contains the 'EUR' currency. Substring
   The 'USDBRL' symbol at index 139 in the list contains the 'USD' currency. Substring
   */
  }
```

Véase también

StringSubstr, StringGetCharacter, StringLen, StringLen

© 2000-2025, MetaQuotes Ltd.

StringGetCharacter

Devuelve el valor de un símbolo que se encuentra en la posición especificada de una cadena.

```
ushort  StringGetCharacter(
   string  string_value,     // cadena
   int     pos               // posición del símbolo en la cadena
   );
```

Parámetros

string_value

[in] Cadena.

pos

[in] Posición del símbolo en la cadena. Puede ser de 0 a StringLen(text) -1.

Valor devuelto

Código del símbolo, o en caso de algún error devuelve 0. Para obtener el código de error hay que llamar a la función GetLastError().

Ejemplo:

```
void OnStart()
  {
//--- borramos todos los comentarios en el gráfico
   Comment("");
//--- declaramos la cadena de la que obtendremos los valores de los códigos de los sí
   string message = "The script demonstrates the operation of the StringGetCharacter()
   int    length  = StringLen(message);
//--- declaramos la variable de cadena a la que añadiremos los símbolos recibidos de l
   string text    = "";
//--- en un ciclo por el tamaño de la cadena de demostración
   for(int i=0; i<length; i++)
     {
      //--- esperamos 1/10 segundos
      Sleep(100);
      //--- obtenemos el carácter de la cadena situado según el índice del ciclo en la
      ushort char_code=StringGetCharacter(message, i);
      //--- añadimos un carácter a la cadena de salida y mostramos la cadena resultant
      text+=ShortToString(char_code);
      Comment(text);
     }
//--- esperamos dos segundos y eliminamos el comentario del gráfico
   Sleep(2000);
   Comment("");

  /*
  Resultado: la línea de demostración aparecerá letra a letra en la pantalla
   The script demonstrates the operation of the StringGetCharacter() function
```

© 2000-2025, MetaQuotes Ltd.

```
   */
   }
```

Véase también

StringSetCharacter, StringBufferLen, StringLen, StringFill, StringInit, StringToCharArray, StringToShortArray

© 2000-2025, MetaQuotes Ltd.

StringInit

Inicializa una cadena con símbolos especificados y proporciona la longitud especificada de una cadena.

```
bool  StringInit(
   string&    string_var,       // cadena para inicialización
   int        new_len=0,        // longitud necesaria después de inicialización
   ushort     character=0       // símbolo con el que se llena la cadena
   );
```

Parámetros

string_var

[in][out] Cadena que tiene que ser inicializada o deinicializada.

new_len=0

[in] Longitud de la cadena después de inicialización. Si la longitud=0, la cadena se deinicializa, es decir, el buffer de la cadena se limpia y la dirección del buffer se pone a cero.

character=0

[in] Símbolo para llenar la cadena.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false. Para obtener el código de error hay que llamar a la función GetLastError().

Nota

Si character=0 y la longitud new_len>0, entonces el buffer de la cadena con longitud indicada será distribuido y llenado con ceros. La longitud de la cadena será igual a cero porque el buffer entero está llenado con terminadores de la cadena.

Ejemplo:

```
void OnStart()
  {
//---
   string str;
   StringInit(str,200,0);
   Print("str = ",str,": StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/* Resultado
   str = : StringBufferLen(str) = 200   StringLen(str) = 0
*/
```

Véase también

StringBufferLen, StringLen

© 2000-2025, MetaQuotes Ltd.

StringLen

Devuelve el número de símbolos de una cadena.

```
int  StringLen(
   string  string_value      // cadena
   );
```

Parámetros

string_value

[in] Cadena para calcular la longitud.

Valor devuelto

Número de símbolos en la cadena sin contar el cero de terminación.

Ejemplo:

```
void OnStart()
  {
//--- determinamos la cadena de prueba
   string text="123456789012345";
//--- obtenemos el número de caracteres en la cadena
   int str_len=StringLen(text);
//--- imprimimos la cadena y el número de caracteres que contiene en el registro
   PrintFormat("The StringLen() function returned the value of %d chars in string: '%s

  /*
  Resultado
   The StringLen() function returned the value of 15 chars in string: '12345678901234
  */
  }
```

Véase también

StringBufferLen, StringTrimLeft, StringTrimRight, StringToCharArray, StringToShortArray

© 2000-2025, MetaQuotes Ltd.

StringSetLength

Establece para una cadena la longitud indicada en símbolos.

```
bool  StringSetLength(
   string&     string_var,       // cadena
   uint        new_length        // nueva longitud de la cadena
   );
```

Parámetros

string_var

[in][out] Cadena para la que debemos indicar la nueva longitud en símbolos.

new_capacity

[in] Longitud de cadena necesaria en símbolos. Si la nueva longitud new_length es menor que la actual, los símbolos que no quepan serán descartados.

Valor retornado

Si se ejecuta con éxito, retornará true, de lo contrario, false. Para obtener el código del error, hay que llamar la función GetLastError().

Observación

La función StringSetLength() no cambia el tamaño del búfer designado para la cadena.

Ejemplo:

```
void OnStart()
  {
//--- determinamos la cadena
   string text="123456789012345";

//--- mostramos en el registro la cadena y su tamaño
   PrintFormat("Before StringSetLength() the string '%s' has a size of %d characters",

//--- reducimos el tamaño de la cadena a 10 caracteres
   StringSetLength(text, 10);

//--- mostramos la cadena modificada como resultado de la operación StringSetLength()
   PrintFormat("After StringSetLength() the string is now '%s', and has a size of %d c

  /*
  Resultado:
   Before StringSetLength() the string '123456789012345' has a size of 15 characters
   After StringSetLength() the string is now '1234567890', and has a size of 10 charac
  */
  }
```

Mire también

StringLen, StringBufferLen, StringReserve StringInit, StringSetCharacter

© 2000-2025, MetaQuotes Ltd.

StringReplace

Reemplaza todas las subcadenas encontradas en una cadena de caracteres por una secuencia de símbolos especificada.

```
int  StringReplace(
   string&         str,              // cadena en la que se realiza el reemplazamiento
   const string    find,             // subcadena que se busca
   const string    replacement       // subcadena que será insertada en las posiciones
   );
```

Parámetros

str

[in][out] Cadena en la que hay que realizar reemplazos.

find

[in] Subcadena que se busca para ser reemplazada.

replacement

[in] Subcadena que será insertada en lugar de la subcadena encontrada.

Valor devuelto

Número de reemplazos realizados en caso de éxito, de lo contrario devuelve -1. Para obtener el código del error, se debe llamara ala función GetLastError().

Nota

Si la función se ha ejecutado con éxito pero no ha sido realizado ningún reemplazamiento (subcadena a reemplazar no encontrada), la función devuelve 0.

El error puede estar en los parámetros *str* o *find* incorrectos (cadena vacía o no inicializada, más detalles en StringInit()). Además, el error puede surgir si no hay suficiente memoria para completar los reemplazamientos.

Ejemplo:

```
  string text="The quick brown fox jumped over the lazy dog.";
  int replaced=StringReplace(text,"quick","slow");
  replaced+=StringReplace(text,"brown","black");
  replaced+=StringReplace(text,"fox","bear");
  Print("Replaced: ", replaced,". Result=",text);

//  Resultado
//  Replaced: 3. Result=The slow black bear jumped over the lazy dog.
//
```

Véase también

StringSetCharacter(), StringSubstr()

© 2000-2025, MetaQuotes Ltd.

StringReserve

Reserva en la memoria para la cadena un búfer del tamaño indicado.

```
bool  StringReserve(
   string&     string_var,       // cadena
   uint        new_capacity      // tamaño del búfer para guardar la cadena
   );
```

Parámetros

string_var

[in][out] Cadena para la que se debe cambiar el tamaño del búfer.

new_capacity

[in] Tamaño necesario del búfer para la cadena. Si el nuevo tamaño de new_capacity es menor que la longitud de la cadena, el tamaño del búfer actual no cambia.

Valor retornado

Si se ejecuta con éxito, retornará true, de lo contrario, false. Para obtener el código del error, hay que llamar la función GetLastError().

Observación

En general, el tamaño de la cadena no es igual al del búfer designado para el guardado de la cadena. Normalmente, al crear la cadena, el búfer para la misma se designa con margen. La función StringReserve() permite gestionar el tamaño del búfer e indicar el tamaño óptimo para futuras operaciones.

A diferencia de StringInit(), la función StringReserve() no cambia el contenido de la cadena y no la rellena con símbolos.

Ejemplo:

```
void OnStart()
  {
   string s;
//--- comprobamos la velocidad de funcionamiento sin uso de StringReserve
   ulong t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_no_reserve=GetMicrosecondCount()-t0;
   s=NULL;
//--- ahora la medimos con el uso de StringReserve
   StringReserve(s,1024 * 3);
   t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_reserve=GetMicrosecondCount()-t0;
//--- comprobamos el tiempo
   Print("Test with StringReserve passed for "+(string)msc_reserve+" msc");
   Print("Test without StringReserve passed for "+(string)msc_no_reserve+" msc");
```

© 2000-2025, MetaQuotes Ltd.

```
/* Resultado
     Test with StringReserve passed for 50 msc
     Test without StringReserve passed for 121 msc
*/
  }
```

Mire también

StringBufferLen, StringSetLength, StringInit, StringSetCharacter

© 2000-2025, MetaQuotes Ltd.

StringSetCharacter

Devuelve copia de una cadena con el valor de símbolo modificado en una posición especificada.

```
bool  StringSetCharacter(
   string&   string_var,       // cadena
   int       pos,              // posición
   ushort    character         // símbolo
   );
```

Parámetros

string_var

[in][out] Cadena.

pos

[in] Posición del símbolo en la cadena. Puede ser de 0 a StringLen(text).

character

[in] Código de símbolo Unicode.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false. Para obtener el código de error hay que llamar a la función GetLastError().

Nota

Si el valor pos es menos que la longitud de la cadena y el valor del código de símbolo = 0, la cadena se recorta (pero el tamaño del buffer distribuido para esta cadena se queda sin cambiar). La longitud de la cadena se iguala a pos.

Si el valor de parámetro *pos* es igual al valor de longitud, el símbolo especificado se añade al final de la cadena, de esta manera la longitud se aumente a uno.

Ejemplo:

```
void OnStart()
  {
   string str="0123456789";
   Print("before: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- introducimos el valor cero entre la cadena
   StringSetCharacter(str,6,0);
   Print("  after: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- añadimos el símbolo al final de la cadena
   int size=StringLen(str);
   StringSetCharacter(str,size,'+');
   Print("addition: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) =",StringLen(str));
  }
/* Resultado
```

© 2000-2025, MetaQuotes Ltd.

```
   before: str = 0123456789 ,StringBufferLen(str) = 0   StringLen(str) = 10
    after: str = 012345 ,StringBufferLen(str) = 16   StringLen(str) = 6
   addition: str = 012345+ ,StringBufferLen(str) = 16   StringLen(str) = 7
*/
```

Véase también

StringBufferLen, StringLen, StringFill, StringInit, CharToString, ShortToString, CharArrayToString, ShortArrayToString

© 2000-2025, MetaQuotes Ltd.

StringSplit

Obtiene las subcadenas desde la cadena especificada según el separador establecido y devuelve el número de subcadenas obtenidas.

```
int  StringSplit(
   const string   string_value,       // cadena para buscar subcadenas
   const ushort   separator,          // separador usando el cual se realizará la búsc
   string         & result[]          // array pasado por referencia para obtener las
   );
```

Parámetros

string_value

[in] La cadena desde la cual hay que obtener las subcadenas. La cadena en sí no se cambia.

pos

[in] Código del símbolo del separador. Para obtener el código, se puede usar la función StringGetCharacter().

result[]

[out] Un array de cadenas en el que se colocan las subcadenas obtenidas.

Valor devuelto

Número de cadenas obtenidas en el array result[]. Si en la cadena pasada no se ha encontrado el separador, entonces en el array se colocará sólo una cadena fuente.

Si la cadena string_value está vacía o NULL, la función devolverá cero. En caso del error la función devolverá -1. Para obtener el código del error, hay que llamar a la función GetLastError().

Ejemplo:

```
string to_split="_vida_es_bella_";// cadena que hay que dividir en subcadenas
   string sep="_";                // separador en forma de un símbolo
   ushort u_sep;                  // código del separador
   string result[];               // array para obtener cadenas
   //--- obtenemos el código del separador
   u_sep=StringGetCharacter(sep,0);
   //--- dividimos la cadena en subcadenas
   int k=StringSplit(to_split,u_sep,result);
   //--- Mostramos el comentario
   PrintFormat("Se ha obtenido cadenas: %d. Se ha utilizado el separador '%s' con el c
   //--- ahora visualizaremos todas las cadenas obtenidas
   if(k>0)
     {
      for(int i=0;i<k;i++)
        {
         PrintFormat("result[%d]=\"%s\"",i,result[i]);
        }
     }
```

Véase también

StringReplace(), StringSubstr(), StringConcatenate()

© 2000-2025, MetaQuotes Ltd.

StringSubstr

Extrae una subcadena de una cadena de texto que se empieza desde una posición especificada.

```
string  StringSubstr(
   string  string_value,      // cadena
   int     start_pos,         // posición de inicio
   int     length=-1          // longitud de la cadena a extraer
   );
```

Parámetros

string_value

[in] Cadena de la que se extrae una subcadena.

start_pos

[in] Posición inicial de subcadena. Puede ser de 0 a StringLen(text) -1.

length=-1

[in] Longitud de la cadena a extraer. Si el valor del parámetro es igual a -1, o el parámetro no está definido, entonces la subcadena será extraida empezando desde la posición especificada hasta el final de la cadena.

Valor devuelto

Si es posible devuelve una copia de subcadena extraida. De lo contrario se devuelve una cadena vacía.

Ejemplo:

```
void OnStart()
  {
//--- obtenemos el nombre del símbolo actual
   string name = Symbol();

//--- obtenemos la divisa básica y la cotizada del símbolo
   string base   = StringSubstr(name, 0, 3);
   string quoted = StringSubstr(name, 3, 3);

//--- imprimimos en el registro las divisas obtenidas del símbolo
   PrintFormat("Symbol: %s. Currency base: %s, currency quoted: %s", name, base, quote

  /*
  Resultado
   Symbol: EURUSD. Currency base: EUR, currency quoted: USD
  */
  }
```

Véase también

StringSplit, StringFind, StringGetCharacter

© 2000-2025, MetaQuotes Ltd.

StringToLower

Transforma todos los símbolos de una cadena indicada en minúsculas en el lugar.

```
bool  StringToLower(
   string&  string_var      // cadena para procesar
   );
```

Parámetros

string_var

[in][out] Cadena.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false. Para obtener el código de error hay que llamar a la función GetLastError().

Ejemplo:

```
void OnStart()
  {
//--- definimos la cadena de origen en mayúsculas
   string text=" - THIS STRING, WRITTEN IN UPPERCASE, MUST BE WRITTEN IN LOWERCASE";
//--- Mostramos la cadena de origen en el registro
   Print("Source line:\n", text);
//--- convertimos todos los caracteres de la cadena a minúsculas y enviamos el resulta
   if(StringToLower(text))
      Print("The original string after using the StringToLower() function:\n", text);

  /*
  Resultado
   Source line:
    - THIS STRING, WRITTEN IN UPPERCASE, MUST BE WRITTEN IN LOWERCASE
   The original string after using the StringToLower() function:
    - this string, written in uppercase, must be written in lowercase
  */
  }
```

Véase también

StringToUpper, StringTrimLeft, StringTrimRight

© 2000-2025, MetaQuotes Ltd.

StringToUpper

Transforma todos los símbolos de una cadena indicada en mayúsculas en el lugar.

```
bool  StringToUpper(
   string&  string_var      // cadena para procesar
   );
```

Parámetros

string_var

[in][out] Cadena.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false. Para obtener el código de error hay que llamar a la función GetLastError().

Ejemplo:

```
void OnStart()
  {
//--- definimos la cadena de origen en minúsculas
   string text=" - this string, written in lowercase, must be written in uppercase";
//--- Mostramos la cadena de origen en el registro
   Print("Source line:\n", text);
//--- convertimos todos los caracteres de la cadena a mayúsculas y enviamos el resulta
   if(StringToUpper(text))
      Print("The original string after using the StringToUpper() function:\n", text);

  /*
   Resultado
   Source line:
    - this string, written in lowercase, must be written in uppercase
   The original string after using the StringToUpper() function:
    - THIS STRING, WRITTEN IN LOWERCASE, MUST BE WRITTEN IN UPPERCASE
  */
  }
```

Véase también

StringToLower, StringTrimLeft, StringTrimRight

© 2000-2025, MetaQuotes Ltd.

StringTrimLeft

Borra los símbolos de salto de línea, espacios y símbolos de tabulación desde el inicio de la cadena hasta el primer símbolo significativo. Cadena se modifica en el lugar.

```
int  StringTrimLeft(
   string&  string_var      // cadena para recortar
   );
```

Parámetros

string_var

[in][out] Cadena que será recortada por la izquierda.

Valor devuelto

Devuelve el número de símbolos cortados.

Ejemplo:

```
void OnStart()
  {
//--- definimos la cadena de origen con seis espacios a la izquierda
   string text="      All spaces on the left will be removed from this string";
//--- Mostramos la cadena de origen en el registro
   PrintFormat("Source line:\n'%s'", text);
//--- eliminamos todos los espacios a la izquierda y mostramos en el registro el núme
   int num=StringTrimLeft(text);
   PrintFormat("The StringTrimLeft() function removed %d chars from the left side. Nov

   /*
   Resultado
   Source line:
   '      All spaces on the left will be removed from this string'
   The StringTrimLeft() function removed 6 chars from the left side. Now the line loo
   'All spaces on the left will be removed from this string'
   */
  }
```

Véase también

StringTrimRight, StringToLower, StringToUpper

© 2000-2025, MetaQuotes Ltd.

StringTrimRight

Borra los símbolos de salto de línea, espacios y símbolos de tabulación desde el último símbolo significativo hasta el final de la cadena. Cadena se modifica en el lugar.

```
int  StringTrimRight(
   string&  string_var      // cadena para recortar
   );
```

Parámetros

string_var

[in][out] Cadena que será recortada por la derecha.

Valor devuelto

Devuelve el número de símbolos cortados.

Ejemplo:

```
void OnStart()
  {
//--- definimos la cadena de origen con seis espacios a la derecha
   string text="All spaces on the right will be removed from this string      ";
//--- Mostramos la cadena de origen en el registro
   PrintFormat("Source line:\n'%s'", text);
//--- eliminamos todos los espacios a la derecha y mostramos en el registro el número
   int num=StringTrimRight(text);
   PrintFormat("The StringTrimRight() function removed %d chars from the right side. N

  /*
  Resultado
   Source line:
   'All spaces on the right will be removed from this string      '
   The StringTrimRight() function removed 6 chars from the right side. Now the line lo
   'All spaces on the right will be removed from this string'
  */
  }
```

Véase también

StringTrimLeft, StringToLower, StringToUpper

© 2000-2025, MetaQuotes Ltd.

Fecha y hora

Es un grupo de funciones que se usan para trabajar con los datos del tipo datetime (el número entero que representa la cantidad de segundos han pasado desde 0 horas del 1 de Enero del año 1970).

Para organizar los contadores y temporizadores de alta resolución, se debe utilizar la función GetTickCount() que ofrece los valores en millisegundos.

Función	Acción
TimeCurrent	Devuelve la última hora conocida del servidor (la hora de llegada de la última cotización) en el formato datetime
TimeTradeServer	Devuelve la hora actual de calculación del servidor comercial
TimeLocal	Devuelve la hora local del ordenador en el formato datetime
TimeGMT	Devuelve la hora GMT en el formato datetime, teniendo en cuenta el cambio de horarios de verano o de invierno, según la hora local del ordenador en el que está funcionando el terminal de cliente
TimeDaylightSavings	Devuelve el indicio de cambio horario verano/invierno
TimeGMTOffset	Devuelve la diferencia actual entre la hora GMT y hora local del ordenador en segundos, teniendo en cuenta el cambio horario verano/invierno
TimeToStruct	Convierte un valor del tipo datetime a una variable del tipo de estructura MqlDateTime
StructToTime	Convierte una variable del tipo de estructura MqlDateTime a un valor del tipo datetime

© 2000-2025, MetaQuotes Ltd.

TimeCurrent

Devuelve la última hora conocida del servidor, la hora de llegada de la última cotización para uno de los símbolos seleccionados de la ventana "Estudio de mercado". En el manejador OnTick() dicha función devolverá la hora del tick procesado que ha llegado. En otras ocasiones (por ejemplo, llamada en los manejadores OnInit(), OnDeinit(), OnTimer(), etc.) es la hora de llegada de la última cotización para cualquier símbolo disponible en la ventana "Estudio de mercado", la misma hora que se muestra en el encabezamiento de esta ventana. El valor de la hora se forma en el servidor comercial y no depende de las configuraciones en el ordenador de usuario. Existen 2 variantes de la función.

Llamada sin parámetros

```
datetime  TimeCurrent();
```

Llamada con el parámetro del tipo MqlDateTime

```
datetime  TimeCurrent(
   MqlDateTime&  dt_struct      // variable del tipo de estructura
   );
```

Parámetros

dt_struct

[out] Variable del tipo de estructura MqlDateTime.

Valor devuelto

Valor del tipo datetime

Nota

Si la variable del tipo de estructura MqlDateTime ha sido pasada como un parámetro, entonces ella se rellena de modo correspondiente.

Para organizar los contadores y temporizadores de alta resolución, se debe utilizar la función GetTickCount() que ofrece los valores en millisegundos.

Durante el trabajo en el Probador de Estrategias, la hora de la última cotización TimeCurrent() se construye de acuerdo con los datos históricos.

Ejemplo:

```
void OnStart()
  {
//--- declaramos la variable MqlDateTime para que se rellene con datos de fecha/hora y
   MqlDateTime tm={};
   datetime    time1=TimeCurrent();    // primera forma de llamada: hora de la última
   datetime    time2=TimeCurrent(tm);  // segunda forma de la llamada: hora de la últi

//--- mostramos el resultado de la obtención de la fecha/hora y rellenamos la estructu
   PrintFormat("Tick time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\n-
               (string)time1, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day
   /*
  Resultado
```

© 2000-2025, MetaQuotes Ltd.

```
   Tick time: 2024.04.18 15:40:06
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 15
   - Min: 40
   - Sec: 06
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeTradeServer

Devuelve la hora actual de calculación del servidor comercial. A diferencia de la función TimeCurrent(), el cálculo del valor de la hora se realiza en el terminal de cliente y depende de las configuraciones de hora en el ordenador de usuario. Existen 2 variantes de la función.

Llamada sin parámetros

```
datetime  TimeTradeServer();
```

Llamada con el parámetro del tipo MqlDateTime

```
datetime  TimeTradeServer(
   MqlDateTime&  dt_struct      // variable del tipo de estructura
   );
```

Parámetros

dt_struct

[out] Variable del tipo de estructura MqlDateTime.

Valor devuelto

Valor del tipo datetime

Nota

Si la variable del tipo de estructura MqlDateTime ha sido pasada como un parámetro, entonces ella se rellena de modo correspondiente.

Para organizar los contadores y temporizadores de alta resolución, se debe utilizar la función GetTickCount() que ofrece los valores en millisegundos.

Durante el trabajo en el Probador de Estrategias, TimeTradeServer() siempre es igual a la hora modelada del servidor TimeCurrent().

Ejemplo:

```
void OnStart()
  {
//--- declaramos la variable MqlDateTime para rellenarla con los datos de fecha/hora y
   MqlDateTime tm={};
   datetime    time_current=TimeCurrent();                  // forma de primera llamad
   datetime    time_server =TimeTradeServer(tm);            // segunda forma de llamad
   int         difference  =int(time_current-time_server);  // diferencia entre Time C

//--- mostramos en el registro la hora de la última cotización y la hora actual estima
   PrintFormat("Time Current: %s\nTime Trade Server: %s\n- Year: %u\n- Month: %02u\n-
               "- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of
               (string)time_current, (string)time_server, tm.year, tm.mon, tm.day, tm.
               EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week), difference);
   /*
  Resultado
   Time Current: 2024.04.18 16:10:14
```

© 2000-2025, MetaQuotes Ltd.

```
   Time Trade Server: 2024.04.18 16:10:15
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 16
   - Min: 10
   - Sec: 15
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   Difference between Time Current and Time Trade Server: -1
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeLocal

Devuelve la hora local del ordenador en el que está funcionando el terminal de cliente. Existen 2 variantes de la función.

Llamada sin parámetros

```
datetime  TimeLocal();
```

Llamada con el parámetro del tipo MqlDateTime

```
datetime  TimeLocal(
   MqlDateTime&  dt_struct      // variable del tipo de estructura
   );
```

Parámetros

dt_struct

[out] Variable del tipo de estructura MqlDateTime.

Valor devuelto

Valor del tipo datetime

Nota

Si la variable del tipo de estructura MqlDateTime ha sido pasada como un parámetro, entonces ella se rellena de modo correspondiente.

Para organizar los contadores y temporizadores de alta resolución, se debe utilizar la función GetTickCount() que ofrece los valores en millisegundos.

Durante el trabajo en el Probador de Estrategias, la hora local TimeLocal() siempre es igual a la hora modelada del servidor TimeCurrent().

Ejemplo:

```
void OnStart()
  {
//--- declaramos la variable MqlDateTime para que contenga los datos sobre la hora loc
   MqlDateTime tm={};
   datetime    time1=TimeLocal();   // primera forma de llamada: hora local del ordena
   datetime    time2=TimeLocal(tm); // segunda forma de llamada: hora local del ordena

//--- mostramos en el registro el resultado de la obtenención de la hora local del orc
   PrintFormat("Local time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\r
               (string)time1, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day
   /*
  Resultado
   Local time: 2024.04.18 19:44:09
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 19
```

© 2000-2025, MetaQuotes Ltd.

```
   - Min: 44
   - Sec: 09
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeGMT

Devuelve la hora GMT que se calcula, teniendo en cuenta el cambio de horarios de verano o de invierno, según la hora local del ordenador en el que está funcionando el terminal de cliente. Existen 2 variantes de la función.

Llamada sin parámetros

```
datetime  TimeGMT();
```

Llamada con el parámetro del tipo MqlDateTime

```
datetime  TimeGMT(
   MqlDateTime&  dt_struct      // variable del tipo de estructura
   );
```

Parámetros

dt_struct

[out] Variable del tipo de estructura MqlDateTime.

Valor devuelto

Valor del tipo datetime

Nota

Si la variable del tipo de estructura MqlDateTime ha sido pasada como un parámetro, entonces ella se rellena de modo correspondiente.

Para organizar los contadores y temporizadores de alta resolución, se debe utilizar la función GetTickCount() que ofrece los valores en millisegundos.

Durante el trabajo en el Probador de Estrategias, la hora TimeGMT() siempre es igual a la hora modelada del servidor TimeTradeServer().

Ejemplo:

```
void OnStart()
  {
//--- declaramos la variable MqlDateTime para llenarla con los datos de fecha/hora y c
   MqlDateTime tm={};
   datetime    time1=TimeLocal();            // primera forma de llamada: hora local c
   datetime    time2=TimeGMT(tm);            // segunda forma de llamada: hora GMT cal
   int         shift=int(time1-time2)/3600;  // desfase de la hora local con respecto

//--- mostramos en el registro la hora local y la hora GMT con los datos de la estruct
   PrintFormat("Time Local: %s\nTime GMT: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n
               "- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of
               (string)time1, (string)time2, tm.year, tm.mon, tm.day, tm.hour, tm.min,
               EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week), shift);
   /*
  Resultado
   Time Local: 2024.04.18 19:37:23
```

© 2000-2025, MetaQuotes Ltd.

```
   Time GMT: 2024.04.18 12:37:23
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 12
   - Min: 37
   - Sec: 23
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   Local time offset relative to GMT: +7
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeDaylightSavings

Devuelve la corrección del horario de verano en segundos, si el cambio al horario de verano ha sido realizado. Depende de la configuración de la hora en el ordenador de usuario.

```
int  TimeDaylightSavings();
```

Valor devuelto

Si ha sido realizado el cambio al horario de invierno (estándar), se devuelve 0.

Ejemplo:

```
void OnStart()
  {
//--- obtenemos la corrección del horario de verano en segundos
   int sec_dl=TimeDaylightSavings();

//--- creamos un texto descriptivo para el valor obtenido
   string text=(sec_dl==0 ? "Standard \"winter\" time is used" :
                StringFormat("Daylight saving time has been switched over. The correct

//--- mostramos la descripción de la corrección del horario de verano en segundos
   Print(text);
   /*
   resultado para el horario de "invierno":
   Standard "winter" time is used

   resultado para el horario de "verano":
   Daylight saving time has been switched over. The correction is -3600 seconds
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeGMTOffset

Devuelve la diferencia actual entre la hora GMT y hora local del ordenador en segundos, teniendo en cuenta el cambio horario verano/invierno. Depende de la configuración de la hora en el ordenador de usuario.

```
int  TimeGMTOffset();
```

Valor devuelto

Valor del tipo int que representa la diferencia actual entre la hora GMT y la hora local del ordenador TimeLocal() en segundos.

```
TimeGMTOffset() =  TimeGMT() - TimeLocal()
```

Ejemplo:

```
void OnStart()
  {
//--- obtenemos la hora local, la hora GMT y la diferencia entre la hora GMT y la hora
   datetime time_local=TimeLocal();
   datetime time_gmt  =TimeGMT();
   int      offset    =TimeGMTOffset();

//--- mostramos en el registro los valores obtenidos de tiempo y desfase
   PrintFormat("Local Time: %s, GMT Time: %s, Seconds Offset: %+d", (string)time_local
   /*
  Resultado
   Local Time: 2024.04.18 19:35:52, GMT Time: 2024.04.18 12:35:52, Seconds Offset: -25
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeToStruct

Convierte un valor del tipo datetime (cantidad de segundos transcurridos desde 01.01.1970) a una variable del tipo de estructura MqlDateTime.

```
bool  TimeToStruct(
   datetime      dt,            // fecha y hora
   MqlDateTime&  dt_struct      // estructura para recibir valores
   );
```

Parámetros

dt

[in] Valor de la fecha para la conversión.

dt_struct

[out] Variable de estructura del tipo MqlDateTime.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Ejemplo:

```
void OnStart()
  {
//---obtenemos la última hora conocida del servidor, declaramos la estructura fecha/ho
   datetime    time=TimeCurrent();
   MqlDateTime tm  ={};
   if(!TimeToStruct(time,tm))
      Print("TimeToStruct() failed. Error ", GetLastError());

//--- registramos la hora del servidor obtenida y el resultado del rellenado de la est
   PrintFormat("Server time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\
               (string)time, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day_
   /*
  Resultado
   Server time: 2024.04.18 15:47:27
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 15
   - Min: 47
   - Sec: 27
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

StructToTime

Convierte una variable del tipo de estructura MqlDateTime a un valor del tipo datetime y devuelve el valor final.

```
datetime  StructToTime(
   MqlDateTime&  dt_struct      // estructura de fecha y hora
   );
```

Parámetros

dt_struct

[in] Variable de estructura del tipo MqlDateTime.

Valor devuelto

Valor del tipo datetime que contiene la cantidad de segundos transcurridos desde 01.01.1970.

Ejemplo:

```
#property script_show_inputs

input int   InpYear  =  0;    // Year
input int   InpMonth =  0;    // Month
input int   InpDay   =  0;    // Day
input int   InpHour  =  0;    // Hour
input int   InpMin   =  0;    // Minutes
input int   InpSec   =  0;    // Seconds

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- corregimos los valores introducidos y los escribimos en las variables
   int year =  (InpYear<1970 ? 1970 : InpYear); // Si el año introducido es inferior a
   int month=  (InpMonth<1 ?  1  :  InpMonth > 12  ?  12 :  InpMonth);
   int day  =  (InpDay  <1 ?  1  :  InpDay   > 31  ?  31 :  InpDay);
   int hour =  (InpHour <0 ?  0  :  InpHour  > 23  ?  23 :  InpHour);
   int min  =  (InpMin  <0 ?  0  :  InpMin   > 59  ?  59 :  InpMin);
   int sec  =  (InpSec  <0 ?  0  :  InpSec   > 59  ?  59 :  InpSec);

//--- mostramos los valores en el diario de registro
   PrintFormat("Entered date and time: %04u.%02u.%02u %02u:%02u:%02u", InpYear, InpMor

//--- mostramos en el diario de registro los valores introducidos corregidos
   PrintFormat("Corrected date and time: %04u.%02u.%02u %02u:%02u:%02u", year, month,

//--- escribir los valores de entrada en los campos correspondientes de la estructura
   MqlDateTime time_struct={};
   time_struct.year= year;
```

© 2000-2025, MetaQuotes Ltd.

```
   time_struct.mon = month;
   time_struct.day = day;
   time_struct.hour= hour;
   time_struct.min = min;
   time_struct.sec = sec;

//--- convertimos la fecha y la hora de la estructura en una variable de tipo datetime
   datetime time = StructToTime(time_struct);

//--- mostramos el resultado de la conversión de una variable de tipo de estructura Mq
   Print("Converted date and time: ",TimeToString(time,TIME_DATE|TIME_MINUTES|TIME_SEC
   /*
  Resultados si introducimos valores por defecto cero:
   Entered date and time: 0000.00.00 00:00:00
   Corrected date and time: 1970.01.01 00:00:00
   Converted date and time: 1970.01.01 00:00:00

  resultados si introducimos un día de febrero-mes del año en curso incorrecto:
   Entered date and time: 2024.02.31 00:00:00
   Corrected date and time: 2024.02.31 00:00:00
   Converted date and time: 2024.03.02 00:00:00
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Información de cuenta

Las funciones que devuelven los parámetros de la cuenta corriente.

Función	Acción
AccountInfoDouble	Devuelve el valor del tipo double de la propiedad correspondiente de la cuenta
AccountInfoInteger	Devuelve el valor del tipo de números enteros (bool,int o long) de la propiedad correspondiente de la cuenta
AccountInfoString	Devuelve el valor del tipo string de la propiedad correspondiente de la cuenta

© 2000-2025, MetaQuotes Ltd.

AccountInfoDouble

Devuelve el valor de la propiedad correspondiente de la cuenta.

```
double  AccountInfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  property_id      // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad. El valor puede ser uno de los valores ENUM_ACCOUNT_INFO_DOUBLE.

Valor devuelto

Valor del tipo double.

Ejemplo:

```
void OnStart()
  {
//--- sacamos toda la información que se pueda de la función AccountInfoDouble()
   printf("ACCOUNT_BALANCE =  %G",AccountInfoDouble(ACCOUNT_BALANCE));
   printf("ACCOUNT_CREDIT =  %G",AccountInfoDouble(ACCOUNT_CREDIT));
   printf("ACCOUNT_PROFIT =  %G",AccountInfoDouble(ACCOUNT_PROFIT));
   printf("ACCOUNT_EQUITY =  %G",AccountInfoDouble(ACCOUNT_EQUITY));
   printf("ACCOUNT_MARGIN =  %G",AccountInfoDouble(ACCOUNT_MARGIN));
   printf("ACCOUNT_MARGIN_FREE =  %G",AccountInfoDouble(ACCOUNT_MARGIN_FREE));
   printf("ACCOUNT_MARGIN_LEVEL =  %G",AccountInfoDouble(ACCOUNT_MARGIN_LEVEL));
   printf("ACCOUNT_MARGIN_SO_CALL = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL));
   printf("ACCOUNT_MARGIN_SO_SO = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_SO));
  }
```

Véase también

SymbolInfoDouble, SymbolInfoString, SymbolInfoInteger, PrintFormat

© 2000-2025, MetaQuotes Ltd.

AccountInfoInteger

Devuelve el valor de la propiedad correspondiente de la cuenta.

```
long  AccountInfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  property_id      // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad. El valor puede ser uno de los valores ENUM_ACCOUNT_INFO_INTEGER.

Valor devuelto

Valor del tipo long.

Nota

La propiedad debe ser uno de los tipos bool, int o long.

Ejemplo:

```
void OnStart()
  {
//--- sacamos toda la información que se pueda de la función AccountInfoInteger()
   printf("ACCOUNT_LOGIN =  %d",AccountInfoInteger(ACCOUNT_LOGIN));
   printf("ACCOUNT_LEVERAGE =  %d",AccountInfoInteger(ACCOUNT_LEVERAGE));
   bool thisAccountTradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_ALLOWED);
   bool EATradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_EXPERT);
   ENUM_ACCOUNT_TRADE_MODE tradeMode=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(ACCOU
   ENUM_ACCOUNT_STOPOUT_MODE stopOutMode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoIntege

//--- avisamos sobre la posibilidad de realizar las operaciones comerciales
   if(thisAccountTradeAllowed)
      Print("La actividad comercial para esta cuenta está permitida");
   else
      Print("La actividad comercial para esta cuenta está prohibida!");

//--- comprobamos si los expertos pueden comerciar en esta cuenta
   if(EATradeAllowed)
      Print("La actividad comercial usando los Asesores Expertos está permitida para e
   else
      Print("La actividad comercial usando los Asesores Expertos está prohibida para e

//--- averiguamos el tipo de cuenta
   switch(tradeMode)
     {
      case(ACCOUNT_TRADE_MODE_DEMO):
         Print("Es una cuenta de demostración");
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
      case(ACCOUNT_TRADE_MODE_CONTEST):
         Print("Es una cuenta de concurso");
         break;
      default:Print("Es una cuenta real!");
     }

//--- averiguamos el régimen de establecimiento del nivel StopOut
   switch(stopOutMode)
     {
      case(ACCOUNT_STOPOUT_MODE_PERCENT):
         Print("El nivel StopOut se especifica en porcentaje");
         break;
      default:Print("El nivel StopOut se especifica en moneda");
     }
  }
```

Véase también

Información de cuenta

© 2000-2025, MetaQuotes Ltd.

AccountInfoString

Devuelve el valor de la propiedad correspondiente de la cuenta.

```
string  AccountInfoString(
   ENUM_ACCOUNT_INFO_STRING  property_id      // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad. El valor puede ser uno de los valores ENUM_ACCOUNT_INFO_STRING.

Valor devuelto

Valor del tipo string.

Ejemplo:

```
void OnStart()
  {
//--- sacamos toda la información que se pueda de la función AccountInfoString()
   Print("Nombre del agente = ",AccountInfoString(ACCOUNT_COMPANY));
   Print("Divisa del depósito = ",AccountInfoString(ACCOUNT_CURRENCY));
   Print("Nombre del cliente = ",AccountInfoString(ACCOUNT_NAME));
   Print("Nombre del servidor comercial = ",AccountInfoString(ACCOUNT_SERVER));
  }
```

Véase también

Información de cuenta

© 2000-2025, MetaQuotes Ltd.

Comprobación de estado

Funciones que devuelven los parámetros del estado actual del terminal de cliente

Función	Acción
GetLastError	Devuelve el valor del último error
IsStopped	Devuelve true, si se ha recibido el comando de terminar la ejecución de un programa mql5
UninitializeReason	Devuelve el código de la causa de deinicialización
TerminalInfoInteger	Devuelve un valor del tipo entero de una propiedad correspondiente del ambiente de programa mql5
TerminalInfoDouble	Devuelve un valor del tipo double de una propiedad correspondiente del ambiente de programa mql5
TerminalInfoString	Devuelve un valor del tipo string de una propiedad correspondiente del ambiente de programa mql5
MQLInfoInteger	Devuelve un valor del tipo entero de una propiedad correspondiente de un programa mql5 en funcionamiento
MQLInfoString	Devuelve un valor del tipo string de una propiedad correspondiente de un programa mql5 en funcionamiento
Symbol	Devuelve el nombre de un símbolo del gráfico corriente
Period	Devuelve el período de tiempo del gráfico corriente
Digits	Devuelve la cantidad de dígitos decimales después del punto que determina la precisión de medición del precio del símbolo del gráfico corriente
Point	Devuelve el tamaño del punto del instrumento actual en divisa de cotización

© 2000-2025, MetaQuotes Ltd.

GetLastError

Devuelve el contenido de la variable de sistema _LastError.

```
int  GetLastError();
```

Valor devuelto

Devuelve el valor del último error ocurrido durante la ejecución de un programa mql5.

Nota

Después de llamar a la función el contenido de la variable _LastError no se pone a cero. Para poner esta variable a cero, hay que llamar a la función ResetLastError()

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlRates rates[1]={};   // aquí anotaremos los datos de la barra actual

//--- llamamos deliberadamente a una función con parámetros inadecuados
   int res=CopyRates(NULL, PERIOD_CURRENT, 0, 2, rates);
   if(res!=2)
      PrintFormat("CopyRates() returned %d. LastError %d", res, GetLastError());

//--- reseteamos el código del último error antes de copiar los datos de la barra actu
   ResetLastError();
//--- si la función funciona incorrectamente, el código de error será diferente de 0
   CopyRates(NULL, PERIOD_CURRENT, 0, 1, rates);
   Print("CopyRates() error ", GetLastError());

//--- imprimimos el array de valores obtenidos
   ArrayPrint(rates);
  }
```

Véase también

Códigos de retorno del servidor comercial

© 2000-2025, MetaQuotes Ltd.

IsStopped

Comprueba el cierre forzoso de un programa mql5.

```
bool  IsStopped();
```

Valor devuelto

Devuelve true, si la variable de sistema _StopFlag contiene un valor distinto a 0. El valor no nulo se escribe en la variable _StopFlag, si ha llegado el comando de terminar la ejecución de un programa mql5. En este caso hay que terminar la ejecución cuanto antes, de lo contrario el programa será cerrado forzosamente desde el exterior dentro de 3 segundos.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- en un ciclo infinito con comprobación de parada
   while(!IsStopped())
     {
      //--- mostramos en el gráfico la hora local de la computadora
      Comment("Time Local: ", TimeToString(TimeLocal(), TIME_DATE|TIME_MINUTES|TIME_SE
      Sleep(16);
     }
   Print("The StopFlag is set. The program will be stopped.");

//--- ponemos orden
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

UninitializeReason

Devuelve el código de la causa de deinicialización.

```
int  UninitializeReason();
```

Valor devuelto

Devuelve el valor de la variable _UninitReason que se forma antes de la llamada a la función OnDeinit(). Este valor depende del motivo que ha provocado la deinicialización.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- obtenemos el código de motivo de desinicialización en la variable de motivo
   int reason=UninitializeReason();
//--- creamos una línea de mensaje con el motivo de la desinicialización y enviamos el
   string message=StringFormat("%s: Uninitialize reason code: %d (%s)",__FUNCTION__, r
   Print(message);
//--- con éxito
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- creamos una línea de mensaje con el código de motivo de desinicialización de la
   string message=StringFormat("%s: Uninitialize reason code: %d (%s)",__FUNCTION__, r
   Print(message);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del motivo de desinicialización           |
//+------------------------------------------------------------------+
string UninitializeReasonDescription(const int reason)
  {
   switch(reason)
     {
      //--- el experto ha detenido su funcionamiento llamando a la función ExpertRemov
      case REASON_PROGRAM :
```

© 2000-2025, MetaQuotes Ltd.

```
        return("Expert Advisor terminated its operation by calling the ExpertRemove()
      //--- programa eliminado del gráfico
      case REASON_REMOVE :
        return("Program has been deleted from the chart");
      //--- programa recompilado
      case REASON_RECOMPILE :
        return("Program has been recompiled");
      //--- el símbolo o periodo del gráfico ha sido modificado
      case REASON_CHARTCHANGE :
        return("Symbol or chart period has been changed");
      //--- el gráfico ha sido cerrado
      case REASON_CHARTCLOSE :
        return("Chart has been closed");
      //--- los parámetros de entrada han sido modificados por el usuario
      case REASON_PARAMETERS :
        return("Input parameters have been changed by a user");
      //--- se ha activizado otra cuenta, o bien ha tenido lugar la reconexión al serv
      case REASON_ACCOUNT :
        return("Another account has been activated or reconnection to the trade server
      //--- se ha aplicado otra plantilla del gráfico
      case REASON_TEMPLATE :
        return("A new template has been applied");
      //--- señal de que el manejador OnInit() ha retornado un valor distinto a cero
      case REASON_INITFAILED :
        return("This value means that OnInit() handler has returned a nonzero value");
      //--- el terminal se ha cerrado
      case REASON_CLOSE :
        return("Terminal has been closed");
     }

//--- motivo de desinicialización desconocido
   return("Unknown reason");
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoInteger

Devuelve el valor de una propiedad correspondiente del ambiente de programa mql5.

```
int  TerminalInfoInteger(
   int  property_id      // identificador de propiedad
   );
```

Parámetros

property_id

[in] Identificador de una propiedad. Puede ser uno de los valores de la enumeración ENUM_TERMINAL_INFO_INTEGER.

Valor devuelto

Valor del tipo int.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el número de build del terminal iniciado y su característica "terminal
   int  build = TerminalInfoInteger(TERMINAL_BUILD);
   bool x64   = TerminalInfoInteger(TERMINAL_X64);

//--- calculamos en el diario los datos obtenidos para el terminal
   PrintFormat("MetaTrader 5 %s build %d", (x64 ? "x64" : "x32"), build);
   /*
   resultado:
   MetaTrader 5 x64 build 4330
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoDouble

Devuelve el valor de la propiedad correspondiente del entorno del programa mql5.

```
double  TerminalInfoDouble(
   int  property_id      // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad. Puede ser uno de los valores de la enumeración ENUM_TERMINAL_INFO_DOUBLE.

Valor devuelto

Valor del tipo double.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el balance del usuario en MQL5.community
   double balance = TerminalInfoDouble(TERMINAL_COMMUNITY_BALANCE);

//--- mostramos el valor obtenido en el diario de registro
   PrintFormat("Balance in MQL5.community: %.2f", balance);
   /*
   resultado
   Balance in MQL5.community: 230.50
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoString

La función devuelve el valor de una propiedad correspondiente del ambiente de programa mql5. La propiedad tiene que ser del tipo string

```
string  TerminalInfoString(
   int  property_id      // identificador de propiedad
   );
```

Parámetros

property_id

[in] Identificador de una propiedad. Puede ser uno de los valores de la enumeración ENUM_TERMINAL_INFO_STRING.

Valor devuelto

Valor del tipo string.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos los datos del sistema operativo y del terminal
   string os_ver  = TerminalInfoString(TERMINAL_OS_VERSION);      // denominación del
   string name    = TerminalInfoString(TERMINAL_NAME);            // nombre del termi
   string path    = TerminalInfoString(TERMINAL_PATH);            // carpeta desde la
   string data    = TerminalInfoString(TERMINAL_DATA_PATH);       // carpeta en la que
   string common  = TerminalInfoString(TERMINAL_COMMONDATA_PATH); // carpeta común de

//--- mostramos los datos obtenidos en el diario de registro
   PrintFormat("OS: %s\nTerminal: %s\n- Path: %s\n- Data path: %s\n- Common Data path:
   /*
   OS: Windows 10 build 19045
   Terminal: MetaTrader 5
   - Path: E:\MetaQuotes\MetaTrader 5
   - Data path: E:\MetaQuotes\MetaTrader 5
   - Common Data path: C:\Users\admin\AppData\Roaming\MetaQuotes\Terminal\Common
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

MQLInfoInteger

Devuelve el valor de una propiedad correspondiente de un programa mql5 en funcionamiento.

```
int  MQLInfoInteger(
   int  property_id      // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad. Puede ser uno de los valores de la enumeración ENUM_MQL_INFO_INTEGER.

Valor devuelto

Valor del tipo int.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos los valores de memoria disponible y consumida para los programas MQL
   int limit = MQLInfoInteger(MQL_MEMORY_LIMIT);   // tamaño máximo posible de la memo
   int used  = MQLInfoInteger(MQL_MEMORY_USED);    // tamaño de la memoria utilizada p

//--- mostramos los datos obtenidos en el diario de rergistro
   PrintFormat("Maximum possible amount of dynamic memory for MQL5 program: %d Mb\n"+
               "Memory used by MQL5 program: %d Mb", limit, used);
   /*
   resultado
   Maximum possible amount of dynamic memory for MQL5 program: 8388608 Mb
   Memory used by MQL5 program: 2 Mb
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

MQLInfoString

Devuelve el valor de una propiedad correspondiente de un programa mql5 en funcionamiento.

```
string  MQLInfoString(
   int  property_id      // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad. Puede ser uno de los valores de la enumeración ENUM_MQL_INFO_STRING.

Valor devuelto

Valor del tipo string.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos los datos del programa iniciado
   string name = MQLInfoString(MQL_PROGRAM_NAME);  // nombre del programa MQL5 iniciad
   string path = MQLInfoString(MQL_PROGRAM_PATH);  // ruta para este programa iniciado

//--- mostramos los datos obtenidos en el diario de registro
   PrintFormat("Name of the running MQL program: '%s'\nPath of the running MQL program
   /*
   resultado:
   Name of the running MQL program: 'MQLInfoString'
   Path of the running MQL program: E:\MetaQuotes\MetaTrader 5\MQL5\Scripts\MQLInfoStr
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Symbol

Devuelve el nombre de un símbolo del gráfico corriente.

```
string  Symbol();
```

Valor devuelto

Valor de la variable de sistema _Symbol, donde se almacena el nombre de un símbolo del gráfico corriente.

Observación

A diferencia de los asesores, indicadores y scripts, los servicios no están vinculados a un gráfico concreto. Por eso, para el servicio Symbol() retornará la línea vacía ("").

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el nombre del símbolo del gráfico actual
   string name = Symbol();

//--- mostramos los datos obtenidos en el diario de registro
   PrintFormat("Current chart symbol name: '%s'", name);
   /*
   resultado
   Current chart symbol name: 'EURUSD'
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Period

Devuelve el período de tiempo del gráfico corriente.

```
ENUM_TIMEFRAMES  Period();
```

Valor devuelto

Contenido de la variable _Period en la que se almacena el valor del período de tiempo del gráfico corriente. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES.

Observación

A diferencia de los asesores, indicadores y scripts, los servicios no están vinculados a un gráfico concreto. Por eso, para el servicio Period() retornará 0.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el valor del marco temporal del gráfico actual y su descripción
   ENUM_TIMEFRAMES period    = Period();
   string          timeframe = StringSubstr(EnumToString(period), 7);

//--- mostramos los datos obtenidos en el diario de registro
   PrintFormat("Current chart timeframe: %s\nTimeframe value: %s (%d)",
               timeframe, EnumToString(period), period);
   /*
   resultado:
   Current chart timeframe: H4
   Timeframe value: PERIOD_H4 (16388)
   */
  }
```

Véase también

PeriodSeconds, Períodos de gráficos, Fecha y hora, Visibilidad de objetos

© 2000-2025, MetaQuotes Ltd.

Digits

Devuelve la cantidad de dígitos decimales después del punto que determina la precisión de medición del precio del símbolo del gráfico corriente.

```
int  Digits();
```

Valor devuelto

Valor de la variable _Digits, en la que se almacena el número de dígitos decimales después del punto que determina la precisión del precio del símbolo del gráfico corriente.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el número de decimales después del punto decimal para el símbolo del g
   int digits = Digits();

//--- mostramos los datos obtenidos en el diario de registro
   Print("Number of decimal digits for the current chart symbol: ", digits);
   /*
   resultado:
   Number of decimal digits for the current chart symbol: 5
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Point

Devuelve el tamaño del punto del símbolo corriente en divisa de cotización.

```
double  Point();
```

Valor devuelto

Valor de la variable _Point que almacena el tamaño del punto del símbolo corriente en divisa de cotización.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el tamaño en puntos del instrumento actual en la divisa de cotización
   double point = Point();

//--- mostramos los datos obtenidos en el diario de registro
   Print("Point size of the current symbol in the quote currency: ", DoubleToString(po
   /*
   resultado:
   Point size of the current symbol in the quote currency: 0.00001
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Procesamiento de eventos

En el lenguaje MQL5 se contempla el procesamiento de ciertos eventos predeterminados. Las funciones para procesar estos eventos deben ser definidas en el programa MQL5: el nombre de la función, el tipo de función retornada, los componentes de los parámetros (si los hay) y sus tipos deberán corresponderse estrictamente con la descripción de la función-manejador del evento.

Dependiendo precisamente del tipo del valor retornado y los tipos de los parámetros, el manejador de eventos del terminal del cliente identifica las funciones que procesan este u otro evento. Si para la función correspondiente se han indicado otros parámetros que no correspondan a las descripciones detalladas más abajo, o se ha indicado otro tipo de valor retornado, esta función no se usará para procesar el evento.

Función	Descripción
OnStart	Se llama en un script al suceder el evento Start para la ejecución de las acciones implementadas en el script
OnInit	Se llama en los indicadores y expertos al suceder el evento Init para inicializar un programa MQL5 en marcha
OnDeinit	Se llama en los indicadores y expertos al suceder el evento Deinit para desinicializar un programa MQL5 en marcha
OnTick	Se llama en los expertos al suceder el evento NewTick para el procesamiento de una nueva cotización
OnCalculate	Se llama en los indicadores al suceder el evento Calculate para el procesamiento del cambio en los datos de precio
OnTimer	Se llama en los indicadores y expertos al suceder el evento periódico Timer, generado por el terminal con el intervalo de precio establecido
OnTrade	Se llama en los expertos al suceder el evento Trade, generado al finalizar una operación comercial en el servidor comercial
OnTradeTransaction	Se llama en los expertos al suceder el evento TradeTransaction para el procesamiento de los resultados de la ejecución de una solicitud comercial
OnBookEvent	Se llama en los expertos al suceder el evento BookEvent para el procesamiento de los cambios en la profundidad de mercado
OnChartEvent	Se llama en los indicadores y expertos al suceder el evento ChartEvent para el procesamiento de los cambios del gráfico provocados por las acciones del usario o el funcionamiento de los programas MQL5
OnTester	Se llama en los expertos al suceder el evento Tester para el procesamiento de las acciones necesarias al finalizar la simulación
OnTesterInit	Se llama en los expertos al suceder el evento TesterInit para el procesamiento de las acciones necesarias antes de comenzar la optimización

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
OnTesterDeinit	Se llama en los expertos al suceder el evento TesterDeinit para la ejecución de las acciones necesarias al finalizar la optimización del experto
OnTesterPass	Se llama en los expertos al suceder el evento TesterPass para el procesamiento de un nuevo frame de datos durante la optimización del experto

El terminal de cliente envía los eventos surgidos a los gráficos abiertos correspondientes. Asimismo, los eventos pueden ser generados por los gráficos(eventos de gráfico) o bien por programas mql5 (eventos personalizados). La generación de eventos de creación y eliminación de objetos gráficos se puede activar y desactivar estableciendo las propiedades del gráfico CHART_EVENT_OBJECT_CREATE y CHART_EVENT_OBJECT_DELETE. Todos los programas mql5 y todos los gráficos tienen su propia cola de eventos, donde se van colocando los eventos que suceden nuevamente.

El programa solo recibe los eventos del gráfico en el que está iniciado. Todos los eventos se procesan uno tras otro según su orden de llegada. Si en la cola ya hay un evento NewTick o bien este evento está siendo procesado, el nuevo evento NewTick no se pondrá en la cola del programa mql5. De manera análoga, si en la cola del programa mql5 ya se encuentra el evento ChartEvent o tal evento está siendo procesado, el nuevo evento de este tipo no se colocará en la cola. El procesamiento de los eventos del temporizador se realiza según el mismo esquema, si en la cola se encuentra o ya se está procesando el evento Timer, el nuevo evento del temporizador no se pondrá en la cola.

Las colas de eventos tienen limitaciones, aunque también tamaño suficiente, por eso es poco probable que un programa que se haya escrito correctamente tenga sobrecarga en la cola. Al sobrecargarse la cola, los nuevos eventos se descartan sin ponerse a la misma.

No se recomienda en ningún caso usar ciclos infinitos para el procesamiento de eventos. La única excepción a esta regla pueden ser los scripts que procesan un solo elemento Start.

Las bibliotecas no procesan ningún evento.

© 2000-2025, MetaQuotes Ltd.

OnStart

Se llama solo en los scripts y servicios al suceder el evento Start. La función ha sido pensada para ejecutar una sola vez las acciones implementadas en el programa. Existen dos variantes de la función.

Versión con retorno del resultado

```
int  OnStart(void);
```

Valor retornado

Valor del tipo int mostrado en la pestaña "Diario de registro".

Después de finalizar el script en el diario de registro se creará una entrada del tipo "script nombre_del_script removed (result code N)", donde N es precisamente el valor que ha retornado la función OnStart().

Después de finalizar el servicio en el diario de registro del terminal se creará una entrada del tipo "service nombre_del_servicio stopped (result code N)", donde N es precisamente el valor que ha retornado la función OnStart().

Es prioritario el uso de la llamada de OnStart() con retorno del resultado de ejecución, ya que este método permite no solo ejecutar un script o servicio, sino también retornar el código de error u otra información útil para analizar el resultado de la ejecución del programa.

La versión sin retorno de resultado se ha dejado solo por compatibilidad con los códigos antiguos. No se recomienda su uso

```
void  OnStart(void);
```

Observación

OnStart() es la única función para procesar los eventos en los scripts y servicios, a estos programas no se envían otros eventos. A su vez, el evento Start no se envía a los expertos e indicadores personalizados.

Ejemplo de script:

```
//--- macros para trabajar con el color
#define XRGB(r,g,b)     (0xFF000000|(uchar(r)<<16)|(uchar(g)<<8)|uchar(b))
#define GETRGB(clr)     ((clr)&0xFFFFFF)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- establecemos el color de la vela descendente
   Comment("Establecemos el color de la vela descendente");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BEAR,GetRandomColor());
   ChartRedraw(); // actualizamos rápidamente el gráfico, sin esperar un nuevo tick
   Sleep(1000);   // hacemos una pausa de 1 segundo, para que podamos ver los cambios
//--- establecemos el color de la vela ascendente
   Comment("Establecemos el color de la vela ascendente");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BULL,GetRandomColor());
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color del fondo
   Comment("Establecemos el color del fondo");
   ChartSetInteger(0,CHART_COLOR_BACKGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color de la línea Ask
   Comment("Establecemos el color de la línea Ask");
   ChartSetInteger(0,CHART_COLOR_ASK,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color de la línea Bid
   Comment("Establecemos el color de la línea Bid");
   ChartSetInteger(0,CHART_COLOR_BID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color de la barra descendente y los cantos de la vela descendent
   Comment("Establecemos el color de la barra descendente y los cantos de la vela desc
   ChartSetInteger(0,CHART_COLOR_CHART_DOWN,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color del gráfico y las velas del tipo "Doji"
   Comment("Establecemos el color del gráfico y las velas del tipo "Doji");
   ChartSetInteger(0,CHART_COLOR_CHART_LINE,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color de la barra ascendente y los cantos de la vela ascendente
   Comment("Establecemos el color de la barra ascendente y los cantos de la vela ascer
   ChartSetInteger(0,CHART_COLOR_CHART_UP,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color de los ejes, las escalas y la línea OHLC
   Comment("Establecemos el color de los ejes, las escalas y la línea OHLC");
   ChartSetInteger(0,CHART_COLOR_FOREGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color de la cuadrícula
   Comment("Establecemos el color de la cuadrícula");
   ChartSetInteger(0,CHART_COLOR_GRID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color del precio Last
   Comment("Establecemos el color del precio Last");
   ChartSetInteger(0,CHART_COLOR_LAST,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color de los niveles de Stop Loss y Take Profit
   Comment("Establecemos el color de los niveles de Stop Loss y Take Profit");
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartSetInteger(0,CHART_COLOR_STOP_LEVEL,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- establecemos el color de los volúmenes y los niveles de apertura de posiciones
   Comment("Establecemos el color de los volúmenes y los niveles de apertura de posici
   ChartSetInteger(0,CHART_COLOR_VOLUME,GetRandomColor());
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Retorna un color generado de forma aleatoria                     |
//+------------------------------------------------------------------+
color GetRandomColor()
  {
   color clr=(color)GETRGB(XRGB(rand()%255,rand()%255,rand()%255));
   return clr;
  }
```

Ver también

Funciones de procesamiento de eventos, Ejecución de programas, Eventos del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

OnInit

Se llama en los indicadores y expertos al suceder el evento Init. La función se ha diseñado para inicializar un programa MQL5 en marcha. Existen dos variantes de la función.

La versión con retorno del resultado

```
int  OnInit(void);
```

Valor retornado

Valor del tipo int, cero designa la inicialización exitosa.

Si se retorna el valor INIT_FAILED, el asesor será descargado del gráfico forzosamente.

Si se retorna el valor INIT_FAILED, el indicador no será descargado del gráfico. En este caso, además, el asesor que quedará en el gráfico no funcionará: el manejador de eventos en el indicador no será llamado.

Es prioritario el uso de la llamada de OnInit() con retorno del resultado de ejecución, ya que este método permite no solo ejecutar la inicialización del programa, sino también retornar el código de error en el caso de que el programa finalice antes de tiempo.

La versión sin retorno de resultado se ha dejado solo por compatibilidad con los códigos antiguos. No se recomienda su uso

```
void  OnInit(void);
```

Observación

El evento Init se genera justo después de cargar el experto o indicador, para los scripts este evento no se genera. La función OnInit() se usa para inicializar un programa MQL5. Si OnInit() tiene un valor retornable del tipo int, entonces un código de retorno distinto a cero significará la desinicialización fallida y generará el evento Deinit con el código del motivo de la desinicialización REASON_INITFAILED.

La función OnInit() del tipo void siempre denota la inicialización exitosa y no se recomienda su uso.

Al optimizar los parámetros de entrada del experto, se recomienda utilizar como código de retorno los valores de la enumeración ENUM_INIT_RETCODE. Estos valores han sido pensados para organizar la gestión del transcurso de la optimización, incluyendo la elección de los agentes de simulación más adecuados. Justo al inicializar el experto, antes de iniciar la propia simulación, podemos solicitar información sobre la configuración y los recursos del agente (número de núcleos, volumen de memoria libre, etc.) con la ayuda de la función TerminalInfoInteger(). Basándonos en la información obtenida, podemos o bien permitir usar este agente de simulación, o bien rechazarlo a la hora de optimizar este experto.

Identificador	Descripción
INIT_SUCCEEDED	La inicialización ha tenido éxito, la simulación del experto puede continuar. Este código significa los mismo que el valor cero, la inicialización del experto en el simulador ha tenido lugar con éxito.

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
INIT_FAILED	Inicialización fallida, no tiene sentido continuar la simulación, debido a errores sin solucionar. Por ejemplo, no se ha logrado crear el indicador necesario para que funcione el experto. El retorno de este valor significa lo mismo que el retorno de un valor distinto a cero, es decir, la inicialización del experto en el simulador ha pasado con éxito.
INIT_PARAMETERS_INCORRECT	Ha sido pensado para que el programador designe correctamente el conjunto de parámetros de entrada; en el recuadro general de optimización, la línea de resultado con este código de retorno se iluminará en color rojo. La simulación para este conjunto de parámetros del experto no se ejecutará, el agente está libre para obtener un nuevo resultado. Al obtener este resultado, el simulador de estrategias con toda garantía no transmitirá este valor a otros agentes para su nueva ejecución.
INIT_AGENT_NOT_SUITABLE	No han aparecido errores en el funcionamiento del programa, pero, por algún motivo, este agente no conviene para realizar la simulación. Por ejemplo, no hay memoria operativa suficiente, no hay soporte de OpenCL, etc. Después de retornar este código, el agente no recibirá más tareas hasta el final de esta optimización.

El uso de OnInit() con retorno de INIT_FAILED/INIT_PARAMETERS_INCORRECT en el simulador tiene una serie de peculiaridades que debemos considerar al optimizar los asesores:

- el conjunto de parámetros para el que OnInit() ha retornado INIT_PARAMETERS_INCORRECT se considera inadecuado para la simulación y no se usará para obtener la próxima población durante la optimización genética. Si estos conjuntos descartados de parámetros son demasiados, este hecho podría provocar que la búsqueda de los parámetros óptimos del asesor arroje resultados incorrectos. El algoritmo de búsqueda presupone que la función de criterio de optimización es suave y no tiene brechas en el conjunto de los parámetros de entrada.
- si OnInit() ha retornado INIT_FAILED, esto significa que la simulación no se puede comenzar, y el asesor es descargado de la memoria del agente. Para ejecutar la siguiente pasada en el nuevo conjunto de parámetros se realizará una nueva carga del asesor. Esto hará que necesitemos mucho más tiempo para iniciar la simulación de la siguiente pasada de optimización, en comparación con la llamada de TesterStop().

Ejemplo de la función OnInit() para el experto

```
//--- input parameters
input int      ma_period=20; // periodo de la media móvil

//--- manejador del indicador usado en el asesor
int indicator_handle;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- comprobamos si el valor ma_period es correcto
   if(ma_period<=0)
     {
      PrintFormat("Valor no permitido para el parámetro de entrada ma_period: %d",ma_p
      return (INIT_PARAMETERS_INCORRECT);
     }
//--- al optimizar
   if(MQLInfoInteger(MQL_OPTIMIZATION))
     {
      //--- comprobamos el volumen de memoria operativa disponible para el agente
      int available_memory_mb=TerminalInfoInteger(TERMINAL_MEMORY_TOTAL);
      if(available_memory_mb<2000)
        {
         PrintFormat("Volumen de memoria insuficiente para el agente de simulación: %d
                     available_memory_mb);
         return (INIT_AGENT_NOT_SUITABLE);
        }
     }
//--- comprobamos la presencia del indicador
   indicator_handle=iCustom(_Symbol,_Period,"My_Indicator",ma_period);
   if(indicator_handle==INVALID_HANDLE)
     {
      PrintFormat("No se ha logrado crear el manejador del indicador My_Indicator. Kod
                  GetLastError());
      return (INIT_FAILED);
     }
//--- la inicialización del experto ha tenido éxito
   return(INIT_SUCCEEDED);
  }
```

Ver también

OnDeinit, TesterHideIndicators, Funciones para el procesamiento de eventos, Ejecución de programas, Eventos del terminal de cliente, Inicialización de variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

OnDeinit

Se llama en los indicadores y expertos al suceder el evento Deinit. La función se ha diseñado para desinicializar un programa MQL5 iniciado.

```
void  OnDeinit(
   const int  reason         // código del motivo de la desinicialización
   );
```

Parámetros

reason

[in] Código del motivo de la desinicialización.

Valor retornado

No hay valor retornado

Observación

El evento Deinit se genera para los expertos e indicadores en los casos siguientes:

- antes de llevar a cabo una desinicialización relacionada con el cambio de símbolo o periodo del gráfico al que está fijado el programa mql5;
- antes de llevar a cabo una desinicialización relacionada con el cambio de los parámetros de entrada;
- antes de iniciar un programa mql5.

El parámetro *reason* puede adoptar los siguientes valores:

Constante	Valor	Descripción
REASON_PROGRAM	0	El experto ha dejado de funcionar, llamando a la función ExpertRemove()
REASON_REMOVE	1	El programa ha sido eliminado del gráfico
REASON_RECOMPILE	2	El programa ha sido recompilado
REASON_CHARTCHANGE	3	El símbolo o periodo del gráfico ha sido modificado
REASON_CHARTCLOSE	4	El gráfico ha sido cerrado
REASON_PARAMETERS	5	Los parámetros de entrada han sido modificados por el usuario
REASON_ACCOUNT	6	Se ha activizado otra cuenta, o bien ha tenido lugar la reconexión al servidor comercial como consecuencia del cambio de ajustes de la cuenta
REASON_TEMPLATE	7	Se ha aplicado otra plantilla del gráfico
REASON_INITFAILED	8	El manejador OnInit() ha retornado un valor distinto a cero
REASON_CLOSE	9	El terminal ha sido cerrado

© 2000-2025, MetaQuotes Ltd.

Podrá obtener los códigos de desinicialización del experto con la función UninitializeReason() o desde la variable predeterminada _UninitReason.

Ejemplo de las funciones OnInit() y OnDeinit() para el experto

```
input int fake_parameter=3;      // parámetro inútil
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Obtenemos el número del build en el que se ha compilado el programa
   Print(__FUNCTION__," Build #",__MQLBUILD__);
//--- También se puede obtener el código del motivo del reinicio en OnInit()
   Print(__FUNCTION__," Cuando el experto se reinicia, podemos obtener el código del r
//--- Primer método para obtener el código del motivo de la desinicialización
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Segundo método para obtener el código del motivo de la desinicialización
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReasc
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Primer método para obtener el código del motivo de la desinicialización
   Print(__FUNCTION__," Código del motivo de la desinicialización = ",reason);
//--- Segundo método para obtener el código del motivo de la desinicialización
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Tercer método para obtener el código del motivo de la desinicialización
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReasc
  }
//+------------------------------------------------------------------+
//| Retorna la descripción de texto del motivo de la desinicialización
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Account was changed";break;
      case REASON_CHARTCHANGE:
         text="Symbol or timeframe was changed";break;
      case REASON_CHARTCLOSE:
         text="Chart was closed";break;
```

© 2000-2025, MetaQuotes Ltd.

```
      case REASON_PARAMETERS:
         text="Input-parameter was changed";break;
      case REASON_RECOMPILE:
         text="Program "+__FILE__+" was recompiled";break;
      case REASON_REMOVE:
         text="Program "+__FILE__+" was removed from chart";break;
      case REASON_TEMPLATE:
         text="New template was applied to chart";break;
      default:text="Another reason";
     }
//---
   return text;
  }
```

Ver también

OnInit, funciones de procesamiento de eventos, Ejecución de programas, Eventos del terminal de cliente, Motivos de la desinicialización, Zona de visibilidad y tiempo de vida útil de las variables, Creación y eliminación de objetos

© 2000-2025, MetaQuotes Ltd.

OnTick

Se llama en los expertos al suceder el evento NewTick para el procesamiento de una nueva cotización.

```
void  OnTick(void);
```

Valor retornado

No hay valor retornado

Observación

El evento NewTick se genera solo para los expertos cuando llega un nuevo tick del símbolo a cuyo gráfico está fijado el experto. Es inútil definir la función OnTick() en un indicador o script personalizado, puesto que el evento NewTick no se genera para ellos.

El evento Tick se genera solo para los expertos, pero esto no significa que los expertos estén obligados a tener la función OnTick(), ya que para los expertos se generan no solo los eventos NewTick, sino también los eventos Timer, BookEvent y ChartEvent.

Todos los eventos se procesan uno tras otro según su orden de llegada. Si en la cola ya hay un evento NewTick o bien este evento está siendo procesado, el nuevo evento NewTick no se pondrá en la cola del programa mql5.

El evento NewTick se genera independientemente de si está permitido o no el comercio automático (botón "Comercio automático"). La prohibición del comercio automático implica solo la prohibición del envío de solicitudes comerciales desde un experto; el funcionamiento del experto, en este caso, no se detendrá.

La prohibición del comercio automático pulsando el botón "Comercio automático" no interrumpe la actual ejecución de la función OnTick().

Ejemplo de experto en el que toda la lógica comercial se encuentra implementada en OnTick()

```
//+------------------------------------------------------------------+
//|                                                   TradeByATR.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Ejemplo de asesor que comercia en la dirección \"explosiva\" de
#property description La vela "\"explosiva\" tiene el cuerpo de un tamaño superior a
#property description "El parámetro \"revers\" invierte la dirección de la señal"

input double lots=0.1;        // volumen en lotes
input double kATR=3;          // longitud de la vela de señal en ATR
input int    ATRperiod=20;    // periodo del indicador ATR
input int    holdbars=8;      // cuántas barras mantenemos la posición
input int    slippage=10;     // Deslizamiento permitido
input bool   revers=false;    // ¿viramos la señal?
input ulong  EXPERT_MAGIC=0;  // MagicNumber del experto
```

© 2000-2025, MetaQuotes Ltd.

```
//--- para guardar el manejador del indicador ATR
int atr_handle;
//--- aquí vamos a guardar los últimos valores de ATR y del cuerpo de la vela
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- inicializamos las variables globales
   last_atr=0;
   last_body=0;
//--- establecemos el volumen correcto
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- creamos el manejador del indicador ATR
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: no se ha logrado crear iATR, código de error %d",__FUNCTION__,G
      return(INIT_FAILED);
     }
//--- el experto se ha inicializado correctamente
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- comunicamos el código de finalización del funcionamiento del experto
   Print(__FILE__,": Código del motivo de la desinicialización = ",reason);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- señal comercial
   static int signal=0; // +1 indica la señal de compra, -1 indica la señal de venta
//--- comprobamos y cerramos las posiciones antiguas, abiertas hace más de "holdbars"
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- comprobamos la aparición de una nueva barra
   if(isNewBar())
     {
      //--- comprobamos la presencia de una señal
      signal=CheckSignal();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- si tenemos abierta una posición de compensación, omitimos la señal hasta que la
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
      signal=0;
      return; // salimos del manejador de eventos NewTick y no entramos en el mercado
     }
//--- para la cuenta de cobertura, cada posición vive y se cierra por separado
   if(signal!=0)
     {
      //--- señal de compra
      if(signal>0)
        {
         PrintFormat("%s: ¡Hay señal de compra! Revers=%s",__FUNCTION__,string(revers)
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- señal de venta
      if(signal<0)
        {
         PrintFormat("%s: ¡Hay señal de venta! Revers=%s",__FUNCTION__,string(revers))
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- final de la función OnTick
  }
//+------------------------------------------------------------------+
//| Comprueba la presencia de una señal comercial                    |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0 indica la ausencia de señal
   int res=0;
//--- obtenemos el valor de ATR en la penúltima barra finalizada (el índice de la bar
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- obtenemos los datos de la última barra cerrada en una matriz del tipo MqlR
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- calculamos el tamaño del cuerpo de la barra en la última barra cerrada
         last_body=bar[0].close-bar[0].open;
         //--- si el cuerpo de la última barra (con índice 1) supera el anterior valo
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // para la vela ascendente el valor es positivo
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      else
         PrintFormat("%s: ¡No se ha podido obtener la última barra! Error",__FUNCTION_
     }
   else
      PrintFormat("%s: ¡No se ha podido obtener el valor del indicador ATR! Error",__F
//--- si está activado el modo de comercio reverso
   res=revers?-res:res;  // si es necesario, viramos la señal (en lugar de 1, retornar
//--- retornamos el valor de la señal comercial
   return (res);
  }
//+------------------------------------------------------------------+
//|  Retorna true al aparecer una nueva barra                        |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // guardamos la hora de apertura de la barra actual
//--- obtenemos la hora de apertura de la barra cero
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- si la hora de apertura ha cambiado, significa que ha aparecido una nueva barra
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- es necesario o no mostrar en el log la información sobre la hora de apertu
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- mostramos el mensaje sobre la hora de apertura de una nueva barra
         PrintFormat("%s: new bar on %s %s opened at %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- obtenemos los datos sobre el último tick
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() failed, error = ",GetLastError());
         //--- mostramos la hora del útlimo tick con una precisión de hasta un miliseg
         PrintFormat("Last tick was at %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- tenemos una nueva barra
      return (true);
     }
//--- no hay barras nuevas
   return (false);
  }
//+------------------------------------------------------------------+
//| Comprar según el mercado con un volumen establecido              |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- compramos al precio de mercado
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Vender según el mercado con un volumen establecido               |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- vendemos al precio de mercado
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Cierre de posiciones según el tiempo de mantenimiento en barras  |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // número de posiciones abiertas
//--- iterar todas las posiciones abiertas
   for(int i=total-1; i>=0; i--)
     {
      //--- parámetros de la posición
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      datetime position_open=(datetime)PositionGetInteger(POSITION_TIME);
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;

      //--- si la posición vive durante bastante tiempo, y además el MagicNumber y el
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
         double volume=PositionGetDouble(POSITION_VOLUME);
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // ponemos el texto en minúsculas para formatear co
         PrintFormat("Cerramos la posición #%I64u %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- estableciendo el tipo de orden y enviando la solicitud comercial
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Preparando y enviando la solicitud comercial                     |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- declarando e inicializando las estructuras
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- parámetros de la solicitud
   request.action   =TRADE_ACTION_DEAL;                     // tipo de operación come
   request.position =pos_ticket;                            // ticket de la posición,
   request.symbol   =Symbol();                              // símbolo
   request.volume   =volume;                                // volumen
   request.type     =type;                                  // tipo de orden
   request.price    =price;                                 // precio de finalización
   request.deviation=slip;                                  // desviación permitida re
   request.magic    =magicnumber;                           // MagicNumber de la order
//--- envío de la solicitud
   if(!OrderSend(request,result))
     {
      //--- mostramos la información sobre el fallo
      PrintFormat("OrderSend %s %s %.2f at %.5f error %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- comunicamos que la operación ha tenido éxito
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Ver también

Funciones de procesamiento de eventos, Ejecución de programas, Eventos del terminal de cliente, OnTimer, OnBookEvent, OnChartEvent

© 2000-2025, MetaQuotes Ltd.

OnCalculate

Se llama en los indicadores al suceder el evento Calculate para el procesamiento del cambio en los datos de precio. Existen dos variantes de la función, no debemos usar ambas dentro de los límites de un mismo indicador.

Cálculo basado en la matriz de datos

```
int  OnCalculate(
   const int        rates_total,       // tamaño de la matriz price[]
   const int        prev_calculated,   // número de barras procesadas en la llamada ar
   const int        begin,             // número de índice en la matriz price[] desde
   const double&    price[]            // matriz de valores para el cálculo
   );
```

Cálculos basados en las series temporales del marco temporal actual

```
int  OnCalculate(
   const int        rates_total,       // tamaño de la series de datos de entrada
   const int        prev_calculated,   // número de barras procesadas en la llamada ar
   const datetime&  time[],            // matriz Time
   const double&    open[],            // matriz Open
   const double&    high[],            // matriz High
   const double&    low[],             // matriz Low
   const double&    close[],           // matriz Close
   const long&      tick_volume[],     // matriz Tick Volume
   const long&      volume[],          // matriz Real Volume
   const int&       spread[]           // matriz Spread
   );
```

Parámetros

rates_total

[in] Tamaño de la matriz price[] o de las series temporales de entrada, disponibles para el indicador para el cálculo. En la segunda variante de la función, el valor del parámetro se corresponde con el número de barras en el gráfico en el que está iniciado.

prev_calculated

[in] Contiene el valor que ha retornado la función OnCalculate() en la llamada anterior. Ha sido pensado para omitir en los cálculos aquellas barras que no han cambiado desde el anterior inicio de esta función.

begin

[in] Valor del índice en la matriz price[], desde el cual comienzan los datos significativos. Permite omitir en los cálculos los datos ausentes o iniciales para los cuales no hay valores correctos.

price[]

[in] Matriz de valores para realizar los cálculos. En calidad de matriz price[] se puede transmitir una de las series temporales de precio, o bien el búfer calculado de algún indicador. El tipo de los datos que han sido transmitidos para el cálculo, se puede conocer con la ayuda de la variable predeterminada _AppliedTo.

© 2000-2025, MetaQuotes Ltd.

time[]

[in] matriz con los valores de tiempo de apertura de las barras.

open[]

[in] Matriz con los valores de los precios de apertura.

high[]

[in] Matriz con los valores de los precios máximos.

low[]

[in] Matriz con los valores de los precios mínimos.

close[]

[in] Matriz con los valores de los precios de cierre.

tick_volume[]

[in] Matriz con los valores de los volúmenes de ticks.

volume[]

[in] Matriz con los valores de los volúmenes comerciales.

spread[]

[in] Matriz con los valores de spread para las barras.

Valor retornado

Valor del tipo int que se transmitirá como parámetro *prev_calculated* en las posteriores llamadas a la función.

Observación

Si la función OnCalculate() retorna un valor cero, en la ventana DataWindow del terminal de cliente no se mostrarán los valores del indicador.

Si desde el momento de la última llamada de la función OnCalculate() han cambiado los datos de precio (se ha cargado una historia más profunda o se han rellanado los huecos de la historia), el valor del parámetro necesario *prev_calculated* será establecido como valor cero por el propio terminal.

Para determinar la dirección de la indexación en las matrices *time[]*, *open[]*, *high[]*, *low[]*, *close[]*, *tick_volume[]*, *volume[]* y *spread[]*, es necesario llamar la función ArrayGetAsSeries(). Para no depender de las características por defecto, es sin duda necesario llamar a ArraySetAsSeries() para aquellas matrices con las que se supone que debemos trabajar.

Al usar la primera variante de la función, el usuario elige la serie temporal o indicador necesarios como matriz price[] en la pestaña Parameters al iniciar el indicador. Para ello, se deberá indicar el elemento necesario en la lista desplegable del campo "Apply to".

Para obtener los valores del indicador personalizado de otros programas mql5, se usa la función iCustom(), que retorna el manejador del indicador para las operaciones sucesivas. En este caso, además, podemos indicar la matriz *price[] necesaria* o el manejador de otro indicador. Este parámetro debe ser transmitido en último lugar en la lista de variables de entrada del indicador personalizado.

© 2000-2025, MetaQuotes Ltd.

Es necesario utilizar la vinculación entre el valor retornado por la función OnCalculate() y el segundo parámetro de entrada *prev_calculated*. El parámetro *prev_calculated*, al llamarse la función, contiene el valor que ha retornado la función OnCalculate() en la llamada anterior. Esto permite implementar algoritmos ahorrativos para calcular el indicador personalizado, evitando así los cálculos repetidos de aquellas barras que no han cambiado desde el anterior inicio de esta función.

Ejemplo de indicador

```
//+------------------------------------------------------------------+
//|                                            OnCalculate_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Ejemplo de cálculo del indicador Momentum"

//---- indicator settings
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_type1   DRAW_LINE
#property indicator_color1  Blue
//---- parámetros de entrada
input int MomentumPeriod=14; // Periodo para el cálculo
//---- búfer de indicador
double    MomentumBuffer[];
//--- variable global para guardar el periodo de los cálculos
int       IntPeriod;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- comprobamos el parámetro de entrada
   if(MomentumPeriod<0)
     {
      IntPeriod=14;
      Print("el parámetro Periodo tiene un valor incorrecto. Para los cálculos se usa
     }
   else
      IntPeriod=MomentumPeriod;
//---- búferes
   SetIndexBuffer(0,MomentumBuffer,INDICATOR_DATA);
//---- nombre del indicador para mostrar en DataWindows y en la subventana
   IndicatorSetString(INDICATOR_SHORTNAME,"Momentum"+"("+string(IntPeriod)+")");
//--- esteblecemos el número de barra desde el que comenzará el dibujado
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,IntPeriod-1);
//--- establecemos 0.0 como valor vacío, es decir, que no se dibuja
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//--- con qué precisión mostrar los valores del indicador
   IndicatorSetInteger(INDICATOR_DIGITS,2);
  }
//+------------------------------------------------------------------+
//|  Cálculo del indicador Momentum                                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,     // tamaño de la matriz price[]
                const int prev_calculated, // cuántas barras se han procesado anterior
                const int begin,           // desde dónde comienzan los datos signific
                const double &price[])     // matriz del valor para el procesamiento
  {
//--- posición inicial para los cálculos
   int StartCalcPosition=(IntPeriod-1)+begin;
//---- si no hay datos suficientes para el cálculo
   if(rates_total<StartCalcPosition)
      return(0);  // salimos con un valor cero, el indicador no ha sido calculado
//--- correct draw begin
   if(begin>0)
      PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,StartCalcPosition+(IntPeriod-1));
//--- comenzamos los cálculos, calculamos la posición de inicio
   int pos=prev_calculated-1;
   if(pos<StartCalcPosition)
      pos=begin+IntPeriod;
//--- ciclo principal de cálculo
   for(int i=pos;i<rates_total && !IsStopped();i++)
      MomentumBuffer[i]=price[i]*100/price[i-IntPeriod];
//--- la ejecución de OnCalculate ha finalizado. Retornamos el nuevo valor de prev_cal
   return(rates_total);
  }
```

Ver también

ArrayGetAsSeries, ArraySetAsSeries, iCustom, funciones de procesamiento de eventos, Ejecución de programas, Eventos del terminal de cliente, Acceso a las series temporales e indicadores

© 2000-2025, MetaQuotes Ltd.

OnTimer

Se llama en los expertos al suceder el evento Timer generado por el terminal con el intervalo establecido.

```
void  OnTimer(void);
```

Valor retornado

No hay valor retornado

Observación

El evento Timer es generado periódicamente por el terminal de cliente para el experto que ha activado el temporizador con la ayuda de la función EventSetTimer(). Normalmente, esta función se llama en la función OnInit(). Al finalizar el funcionamiento del experto es necesario eliminar el temporizador creado usando EventKillTimer(), que normalmente se llama en la función OnDeinit().

Todos los expertos e indicadores trabajan con su propio temporizador y reciben los eventos solo del mismo. Al finalizar el funcionamiento de un programa MQL5, el temporizador es eliminado forzosamente, si este ha sido creado y no ha sido desactivado por la función EventKillTimer().

Si es necesario recibir los eventos del temporizador con mayor frecuencia que una vez por segundo, use EventSetMillisecondTimer() para crear un temporizador de alta resolución.

En general, al dismuinuir el periodo del temporizador, se incrementa el tiempo de simulación, ya que aumenta el número de llamadas del manejador de eventos del temporizador. Al trabajar en el modo de tiempo real, los eventos del temporizador no se generan con una frecuencia mayor a 1 vez cada 10-16 milisegundos: esto se relaciona con las limitaciones del hardware.

Para cada programa no se puede iniciar más de un temporizador. Todos los programas mql5 y todos los gráficos tienen su propia cola de eventos, donde se van colocando los eventos que suceden nuevamente. Si en la cola ya hay un evento Timer o bien este evento está siendo procesado, el nuevo evento Timer no se pondrá en la cola del programa mql5.

Ejemplo de un experto con el manejador OnTimer()

```
//+------------------------------------------------------------------+
//|                                                OnTimer_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Ejemplo de uso de un temporizador para calcular la hora de un s
#property description "Es mejor iniciar el asesor al final de la semana comercial, ant
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- creamos un temporizador con un periodo de 1 segundo
```

© 2000-2025, MetaQuotes Ltd.

```
   EventSetTimer(1);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- eliminamos el temporizador al finalizar el trabajo
   EventKillTimer();

  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- hora de la primera llamada de OnTimer()
   static datetime start_time=TimeCurrent();
//--- hora del servidor comercial en la primera llamada de OnTimer();
   static datetime start_tradeserver_time=0;
//--- hora calculada del servidor comercial
   static datetime calculated_server_time=0;
//--- hora local en la computadora
   datetime local_time=TimeLocal();
//--- hora calculada actual del servidor comercial
   datetime trade_server_time=TimeTradeServer();
//--- si por algún motivo se desconoce la hora del servidor, salimos antes de tiempo
   if(trade_server_time==0)
      return;
//--- si el valor inicial del servidor comercial no se ha establecido aún
   if(start_tradeserver_time==0)
     {
      start_tradeserver_time=trade_server_time;
      //--- establecemos el valor a calcular del servidor comercial
      Print(trade_server_time);
      calculated_server_time=trade_server_time;
     }
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- aumentamos la hora de la primera llamada de OnTimer()
       if(start_tradeserver_time!=0)
          calculated_server_time=calculated_server_time+1;;
      }
//---
   string com=StringFormat("                  Start time: %s\r\n",TimeToString(start_t
   com=com+StringFormat("                   Local time: %s\r\n",TimeToString(local_time
   com=com+StringFormat("TimeTradeServer time: %s\r\n",TimeToString(trade_server_time,
   com=com+StringFormat(" EstimatedServer time: %s\r\n",TimeToString(calculated_server
//--- mostramos los valores de todos los contadores en el gráfico
   Comment(com);
  }
```

Ver también

EventSetTimer, EventSetMillisecondTimer, EventKillTimer, GetTickCount, GetMicrosecondCount, Eventos del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

OnTrade

Se llama en los expertos al suceder el evento Trade. La función ha sido pensada para procesar los cambios en las listas de órdenes, posiciones y transacciones.

```
void  OnTrade(void);
```

Valor retornado

No hay valor retornado

Observación

OnTrade() se llama solo para los expertos: en los indicadores y scripts se ignora, incluso si añade a estos una función con tal nombre y tipo.

Al darse cualquier acción comercial (colocar una orden pendiente, abrir/cerrar una posición, colocar stops, activarse órdenes pendientes, etc.) la historia de órdenes y transacciones y/o la lista de posiciones y órdenes actuales cambiarán de la forma correspondiente.

En el momento del procesamiento de la orden, el servidor comercial envía al terminal un mensaje sobre la llegada del evento comercial Trade. Para recibir de la historia información actual sobre las órdenes y transacciones, es necesario ejecutar de manera preliminar la solicitud de la historia comercial con la ayuda de HistorySelect().

Los eventos comerciales son generados por el servidor en los siguientes casos:

- cambio en las órdenes activas,
- cambios en las posiciones,
- cambios en las transacciones,
- cambios en la historia comercial.

Cada evento Trade puede ser resultado de una o varias solicitudes comerciales. Las solicitudes comerciales son enviadas al servidor con la ayuda de OrderSend() o OrderSendAsync(). Cada solicitud puede generar varios eventos comerciales. No debemos confiar en la regla: "Una solicitud - Un evento Trade", ya que el procesamiento de solicitudes puede tener lugar en varias etapas y cada operación puede cambiar el estado de las órdenes, posiciones e historia comercial.

El manejador OnTrade() se llama después de ejecutar las llamadas correspondientes de OnTradeTransaction(). En general, no existe una proporción exacta en cuanto a la cantidad de llamadas de OnTrade() y OnTradeTransaction(). Una llamada de OnTrade() corresponde a una o varias llamadas de OnTradeTransaction.

Ejemplo de un experto con el manejador OnTrade()

```
//+------------------------------------------------------------------+
//|                                                OnTrade_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
input    int days=7;             // profundidad de la historia comercial en días
//--- establecemos a nivel global los límites de la historia comercial
datetime     start;              // fecha de comienzo de la historia comercial en la ca
datetime     end;                // fecha de finalización de la historia comercial en l
//--- contadores globales
int          orders;             // número de órdenes activas
int          positions;          // número de posiciones abiertas
int          deals;              // número de transacciones en la caché de la historia
int          history_orders;     // número de órdenes en la caché de la historia comerc
bool         started=false;      // bandera de relevancia de los contadores

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   end=TimeCurrent();
   start=end-days*PeriodSeconds(PERIOD_D1);
   PrintFormat("Límites de la historia comercial a cargar: comienzo - %s, final - %s",
               TimeToString(start),TimeToString(end));
   InitCounters();
//---
   return(0);
  }
//+------------------------------------------------------------------+
//|  Inicialización de los contadores de posiciones, órdenes y transacciones
//+------------------------------------------------------------------+
void InitCounters()
  {
   ResetLastError();
//--- cargamos la historia
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. No se ha logrado cargar en la caché la historia desde %s hasta
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- obtenemos los valores actuales
   orders=OrdersTotal();
   positions=PositionsTotal();
   deals=HistoryDealsTotal();
   history_orders=HistoryOrdersTotal();
   started=true;
   Print("Los contadores de posiciones, órdenes y transacciones se han inicializado co
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| se llama al suceder el evento Trade                              |
//+------------------------------------------------------------------+
void OnTrade()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| ejemplo de procesamiento de los cambios en el comercio y la historia
//+------------------------------------------------------------------+
void SimpleTradeProcessor()
  {
   end=TimeCurrent();
   ResetLastError();
//--- cargamos en la caché del programa la historia comercial del intervalo indicado
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. No se ha logrado cargar en la caché la historia desde %s hasta
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- obtenemos los valores actuales
   int curr_orders=OrdersTotal();
   int curr_positions=PositionsTotal();
   int curr_deals=HistoryDealsTotal();
   int curr_history_orders=HistoryOrdersTotal();
//--- comprobamos los cambios en el número de órdenes activas
   if(curr_orders!=orders)
     {
      //--- el número de órdenes activas ha cambiado
      PrintFormat("El número de órdenes ha cambiado. Era %d, ahora es %d",
                  orders,curr_orders);
      //--- actualizamos el valor
      orders=curr_orders;
     }
//--- cambios en el número de posiciones abiertas
   if(curr_positions!=positions)
     {
      //--- el número de posiciones abiertas ha cambiado
      PrintFormat("El número de posiciones ha cambiado. Era %d, ahora es %d",
```

© 2000-2025, MetaQuotes Ltd.

```
                  positions,curr_positions);
      //--- actualizamos el valor
      positions=curr_positions;
     }
//--- cambios en el número de transacciones en la caché de la historia comercial
   if(curr_deals!=deals)
     {
      //--- el número de transacciones en la caché de la historia comercial ha cambiad
      PrintFormat("Ha cambiado el número de transacciones. Era %d, ahora es %d",
                  deals,curr_deals);
      //--- actualizamos el valor
      deals=curr_deals;
     }
//--- cambios en el número de órdenes históricas en la caché de la historia comercial
   if(curr_history_orders!=history_orders)
     {
      //--- el número de órdenes históricas en la caché de la historia comercial ha ca
      PrintFormat("Ha cambiado el número de órdenes en la historia. Era %d, ahora es %
                  history_orders,curr_history_orders);
     //--- actualizamos el valor
     history_orders=curr_history_orders;
     }
//--- comprobamos la necesidad de cambiar los límites de la historia comercial para la
   CheckStartDateInTradeHistory();
  }
//+------------------------------------------------------------------+
//|  cambios de la fecha inicial para solicitar la historia comercial|
//+------------------------------------------------------------------+
void CheckStartDateInTradeHistory()
  {
//--- intervalo inicial, si empezáramos el trabajo ahora mismo
   datetime curr_start=TimeCurrent()-days*PeriodSeconds(PERIOD_D1);
//--- comprobamos que el límite de inicio de la historia comercial se haya apartado no
//--- de 1 día de la fecha pensada
   if(curr_start-start>PeriodSeconds(PERIOD_D1))
     {
      //--- se deberá corregir la fecha de inicio de la historia cargada en la caché
      start=curr_start;
      PrintFormat("Nuevo límite de comienzo de la historia comercial a cargar: inicio
                  TimeToString(start));
      //--- ahora cargamos de nuevo la historia comercial para el intervalo actualizad
      HistorySelect(start,end);
      //--- corregimos los contadores de transacciones y órdenes en la historia para l
      history_orders=HistoryOrdersTotal();
      deals=HistoryDealsTotal();
     }
  }
//+------------------------------------------------------------------+
/* Ejemplo de muestra:
```

© 2000-2025, MetaQuotes Ltd.

```
  Límites de la historia comercial cargada: inicio - 2018.07.16 18:11, final - 2018.07
  Los contadores de órdenes, posiciones y transacciones han sido inicializados correct
  Ha cambiado el número de órdenes. Era 0, ahora es 1
  Ha cambiado el número de órdenes. Era 1, ahora es 0
  Ha cambiado el número de posiciones. Era 0, ahora es 1
  Ha cambiado el número de transacciones. Era 0, ahora es 1
  Ha cambiado el número de órdenes en la historia. Era 0, ahora es 1
*/
```

Ver también

OrderSend, OrderSendAsync, OnTradeTransaction, Eventos del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

OnTradeTransaction

Se llama en los expertos al suceder el evento TradeTransaction. La función ha sido pensada para procesar los resultados de la ejecución de una solicitud comercial.

```
void  OnTradeTransaction()
   const MqlTradeTransaction&    trans,     // estructura de la transacción comercial
   const MqlTradeRequest&        request,   // estructura de la solicitud
   const MqlTradeResult&         result     // estructura de la respuesta
   );
```

Parámetros

trans

[in] Variable del tipo MqlTradeTransaction que describe una transacción realizada en la cuenta comercial.

request

[in] Variable del tipo MqlTradeRequest que describe la solicitud comercial que ha generado la transacción. Contiene valores solo para una transacción del tipo TRADE_TRANSACTION_REQUEST.

result

[in] Variable del tipo MqlTradeResult con el resultado de la ejecución de la solicitud comercial que ha generado la transacción. Contiene valores solo para una transacción del tipo TRADE_TRANSACTION_REQUEST.

Valor retornado

No hay valor retornado

Observación

OnTradeTransaction() se llama para procesar el evento TradeTransaction, que el servidor comercial envía al terminal en los siguientes casos:

- el envío de una solicitud comercial desde un programa MQL5 con la ayuda de las funciones OrderSend()/OrderSendAsync() y su posterior ejecución;
- el envío manual de una solicitud comercial a través de la interfaz gráfica y su posterior ejecución;
- la activación de órdenes pendientes y órdenes stop en el servidor;
- la ejecución de operaciones en lado del servidor comercial.

La información sobre el tipo de información se contiene en el campo *type* de la variable *trans*. Los tipos de transacciones comerciales se describen en la enumeración ENUM_TRADE_TRANSACTION_TYPE:

- TRADE_TRANSACTION_ORDER_ADD - añadir una nueva orden activa
- TRADE_TRANSACTION_ORDER_UPDATE - cambiar una orden activa
- TRADE_TRANSACTION_ORDER_DELETE - eliminar una orden de la lista de activas
- TRADE_TRANSACTION_DEAL_ADD - añadir una transacción a la historia
- TRADE_TRANSACTION_DEAL_UPDATE - cambiar una transacción en la historia
- TRADE_TRANSACTION_DEAL_DELETE - eliminar una transacción de la historia
- TRADE_TRANSACTION_HISTORY_ADD - añadir una orden a la historia como resultado de una ejecución o cancelación

© 2000-2025, MetaQuotes Ltd.

- TRADE_TRANSACTION_HISTORY_UPDATE - cambiar una orden que se encuentra en la historia de órdenes
- TRADE_TRANSACTION_HISTORY_DELETE - eliminar una orden de la historia de órdenes
- TRADE_TRANSACTION_POSITION - cambiar una posición no relacionada con la ejecución de una transacción
- TRADE_TRANSACTION_REQUEST - notificar que la solicitud comercial ha sido procesada por el servidor y que el resultado del procesamiento ha sido recibido.

Al procesar transacciones del tipo TRADE_TRANSACTION_REQUEST, para obtener información adicional es necesario analizar el segundo y tercer parámetros de la función OnTradeTransaction(): *request* y *result*.

El envío de una solicitud comercial de compra genera una cadena de transacciones comerciales que se ejecutan en la cuenta comercial: 1) la solicitud es aceptada para su procesamiento, 2) a continuación, para la cuenta se crea la orden de compra correspondiente, 3) después tiene lugar la ejecución de la orden, 4) se elimina la orden ejecutada de la lista de activas, 5) se añade a la historia de órdenes, 6) a continuación, se añade la transacción correspondiente a la historia y 7) se crea una nueva posición. Todas estas acciones son transacciones comerciales. La llegada de cada transacción semejante al terminal constituye el evento TradeTransaction. En este caso, además, la llegada de estas transacciones al terminal no está garantizada, por eso no debemos construir nuestro algoritmo esperando que los grupos de transacciones comerciales lleguen unos detrás de otros.

Mientras el experto procesa las transacciones comerciales con la ayuda del manejador OnTradeTransaction(), el terminal continúa procesando las transacciones comerciales entrantes. De esta forma, el estado de la cuenta comercial puede cambiar ya durante el funcionamiento de OnTradeTransaction(). Por ejemplo, mientras un programa MQL5 procesa el evento de adición de una nueva orden, esta puede ser ejeuctada, eliminada de la lista de abiertas y trasladada a la historia. Posteriormente, al programa se le notificarán todos estos eventos.

La longitud de la cola de transacciones es de 1024 elementos. Si OnTradeTransaction() procesa la siguiente transacción durante demasiado tiempo, las transacciones antiguas que se encuentren en la cola podrían ser remplazadas por otras nuevas.

El manejador OnTrade() se llama después de las llamadas correspondientes de OnTradeTransaction(). En general, no existe una proporción exacta en cuanto a la cantidad de llamadas de OnTrade() y OnTradeTransaction(). Una llamada de OnTrade() corresponde a una o varias llamadas de OnTradeTransaction.

Cada evento Trade puede ser el resultado de una o varias solicitudes comerciales. Las solicitudes comerciales se envían al servidor con la ayuda de OrderSend() o OrderSendAsync(). Cada solicitud puede generar varios eventos comerciales. No debemos confiar en la regla: "Una solicitud - Un evento Trade", ya que el procesamiento de solicitudes puede tener lugar en varias etapas y cada operación puede cambiar el estado de las órdenes, posiciones e historia comercial.

Ejemplo de un experto con el manejador OnTradeTransaction()

```
//+------------------------------------------------------------------+
//|                                    OnTradeTransaction_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version   "1.00"
#property description "Ejemplo de oyente de eventos TradeTransaction"
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   PrintFormat("LAST PING=%.f ms",
               TerminalInfoInteger(TERMINAL_PING_LAST)/1000.);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//---
   static int counter=0;   // contador de llamadas de OnTradeTransaction()
   static uint lasttime=0; // hora de la última llamada de OnTradeTransaction()
//---
   uint time=GetTickCount();
//--- si la última transacción ha sido hace más de 1 segundo
   if(time-lasttime>1000)
     {
      counter=0; // esto significa que se trata de una nueva operación comercial y poc
      if(IS_DEBUG_MODE)
         Print(" Nueva operación comercial");
     }
   lasttime=time;
   counter++;
   Print(counter,". ",__FUNCTION__);
//--- resultado de la ejecución de la solicitud comercial
   ulong             lastOrderID   =trans.order;
   ENUM_ORDER_TYPE   lastOrderType =trans.order_type;
   ENUM_ORDER_STATE lastOrderState=trans.order_state;
//--- nombre del símbolo para el que se ha ejecuctado la transacción
   string trans_symbol=trans.symbol;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- tipo de transacción
   ENUM_TRADE_TRANSACTION_TYPE  trans_type=trans.type;
   switch(trans.type)
     {
      case  TRADE_TRANSACTION_POSITION:    // modificación de la posición
        {
         ulong pos_ID=trans.position;
         PrintFormat("MqlTradeTransaction: Position  #%I64u %s modified: SL=%.5f TP=%.
                     pos_ID,trans_symbol,trans.price_sl,trans.price_tp);
        }
      break;
      case TRADE_TRANSACTION_REQUEST:      // enviar solicitud comercial
         PrintFormat("MqlTradeTransaction: TRADE_TRANSACTION_REQUEST");
         break;
      case TRADE_TRANSACTION_DEAL_ADD:     // añadir transacción
        {
         ulong           lastDealID   =trans.deal;
         ENUM_DEAL_TYPE lastDealType =trans.deal_type;
         double         lastDealVolume=trans.volume;
         //--- identificador de la transacción en un sistema externo, es el ticket asi
         string Exchange_ticket="";
         if(HistoryDealSelect(lastDealID))
            Exchange_ticket=HistoryDealGetString(lastDealID,DEAL_EXTERNAL_ID);
         if(Exchange_ticket!="")
            Exchange_ticket=StringFormat("(Exchange deal=%s)",Exchange_ticket);

         PrintFormat("MqlTradeTransaction: %s deal #%I64u %s %s %.2f lot   %s",EnumToS
                     lastDealID,EnumToString(lastDealType),trans_symbol,lastDealVolume
        }
      break;
      case TRADE_TRANSACTION_HISTORY_ADD: // añadir una orden a la historia
        {
         //--- identificador de la orden en un sistema externo, es el ticket asignado
         string Exchange_ticket="";
         if(lastOrderState==ORDER_STATE_FILLED)
           {
            if(HistoryOrderSelect(lastOrderID))
               Exchange_ticket=HistoryOrderGetString(lastOrderID,ORDER_EXTERNAL_ID);
            if(Exchange_ticket!="")
               Exchange_ticket=StringFormat("(Exchange ticket=%s)",Exchange_ticket);
           }
         PrintFormat("MqlTradeTransaction: %s order #%I64u %s %s %s   %s",EnumToStringC
                     lastOrderID,EnumToString(lastOrderType),trans_symbol,EnumToStringC
        }
      break;
      default: // otras transacciones
        {
         //--- identificador de la orden en un sistema externo, es el ticket asignado
         string Exchange_ticket="";
```

© 2000-2025, MetaQuotes Ltd.

```
         if(lastOrderState==ORDER_STATE_PLACED)
           {
            if(OrderSelect(lastOrderID))
               Exchange_ticket=OrderGetString(ORDER_EXTERNAL_ID);
            if(Exchange_ticket!="")
               Exchange_ticket=StringFormat("Exchange ticket=%s",Exchange_ticket);
           }
         PrintFormat("MqlTradeTransaction: %s order #%I64u %s %s   %s",EnumToString(tr
                     lastOrderID,EnumToString(lastOrderType),EnumToString(lastOrderSta
        }
      break;
     }
//--- ticket de la orden
   ulong orderID_result=result.order;
   string retcode_result=GetRetcodeID(result.retcode);
   if(orderID_result!=0)
      PrintFormat("MqlTradeResult: order #%d retcode=%s ",orderID_result,retcode_resul
//---
  }
//+------------------------------------------------------------------+
//| convierte los códigos numéricos de respuesta en códigos mnemónicos
//+------------------------------------------------------------------+
string GetRetcodeID(int retcode)
  {
   switch(retcode)
     {
      case 10004: return("TRADE_RETCODE_REQUOTE");             break;
      case 10006: return("TRADE_RETCODE_REJECT");              break;
      case 10007: return("TRADE_RETCODE_CANCEL");              break;
      case 10008: return("TRADE_RETCODE_PLACED");              break;
      case 10009: return("TRADE_RETCODE_DONE");                break;
      case 10010: return("TRADE_RETCODE_DONE_PARTIAL");        break;
      case 10011: return("TRADE_RETCODE_ERROR");               break;
      case 10012: return("TRADE_RETCODE_TIMEOUT");             break;
      case 10013: return("TRADE_RETCODE_INVALID");             break;
      case 10014: return("TRADE_RETCODE_INVALID_VOLUME");      break;
      case 10015: return("TRADE_RETCODE_INVALID_PRICE");       break;
      case 10016: return("TRADE_RETCODE_INVALID_STOPS");       break;
      case 10017: return("TRADE_RETCODE_TRADE_DISABLED");      break;
      case 10018: return("TRADE_RETCODE_MARKET_CLOSED");       break;
      case 10019: return("TRADE_RETCODE_NO_MONEY");            break;
      case 10020: return("TRADE_RETCODE_PRICE_CHANGED");       break;
      case 10021: return("TRADE_RETCODE_PRICE_OFF");           break;
      case 10022: return("TRADE_RETCODE_INVALID_EXPIRATION");  break;
      case 10023: return("TRADE_RETCODE_ORDER_CHANGED");       break;
      case 10024: return("TRADE_RETCODE_TOO_MANY_REQUESTS");   break;
      case 10025: return("TRADE_RETCODE_NO_CHANGES");          break;
      case 10026: return("TRADE_RETCODE_SERVER_DISABLES_AT");  break;
      case 10027: return("TRADE_RETCODE_CLIENT_DISABLES_AT");  break;
```

© 2000-2025, MetaQuotes Ltd.

```
      case 10028: return("TRADE_RETCODE_LOCKED");              break;
      case 10029: return("TRADE_RETCODE_FROZEN");              break;
      case 10030: return("TRADE_RETCODE_INVALID_FILL");        break;
      case 10031: return("TRADE_RETCODE_CONNECTION");          break;
      case 10032: return("TRADE_RETCODE_ONLY_REAL");           break;
      case 10033: return("TRADE_RETCODE_LIMIT_ORDERS");        break;
      case 10034: return("TRADE_RETCODE_LIMIT_VOLUME");        break;
      case 10035: return("TRADE_RETCODE_INVALID_ORDER");       break;
      case 10036: return("TRADE_RETCODE_POSITION_CLOSED");     break;
      default:
         return("TRADE_RETCODE_UNKNOWN="+IntegerToString(retcode));
         break;
     }
//---
  }
```

Ver también

OrderSend, OrderSendAsync, OnTradeTransaction, Estructura de la solicitud comercial, Estructura de la transacción comercial, Tipos de transacciones comerciales, Tipos de operaciones comerciales, Eventos del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

OnBookEvent

Se llama en los indicadores y expertos al suceder el evento BookEvent. La función está pensada para el procesamiento de los cambios de la profundidad del mercado (Depth of Market).

```
void  OnBookEvent(
   const string&  symbol         // símbolo
   );
```

Parámetros

symbol

[in] Nombre del instrumento financiero según el cual ha tenido lugar el evento BookEvent

Valor retornado

No hay valor retornado

Observación

Para obtener el evento BookEvent de cualquier símbolo, basta con suscribirse preliminarmente para recibir estos eventos para este símbolo con la ayuda de la función MarketBookAdd(). Si queremos anular la suscripción para recibir el evento BookEvent de un símbolo concreto, debemos llamar la función MarketBookRelease().

El evento BookEvent retransmite dentro de los límites del gráfico. Esto significa que basta con que una aplicación en el gráfico se suscriba para recibir el evento BookEvent con la ayuda de la función MarketBookAdd, y todos los demás indicadores y expertos iniciados en este gráfico y que además tengan el manejador OnBookEvent() recibirán este evento. Por eso mismo, es necesario analizar el nombre del símbolo que se transmite al manejador OnBookEvent() como parámetro *symbol*.

Para todas las aplicaciones iniciadas en un gráfico se dispone de contadores aparte para recibir los eventos BookEvent para cada símbolo por separado. Esto significa que en cada gráfico puede haber varias suscripciones a varios símbolos, y para cada símbolo hay un contador propio. La suscripción y la baja de la suscripción de los eventos de BookEvent solo cambian el contador de suscripciones de los símbolos indicados, pero en este caso, además, solo dentro de los límites del gráfico. Esto significa que en dos gráficos colindantes puede haber suscripciones a los eventos BookEvent en el mismo símbolo, pero con diferentes valores en los contadores de suscripciones.

El valor inicial del contador de suscripciones es igual a cero. Con cada llamada de MarketBookAdd(), el contador de suscripciones para el símbolo indicado en cada gráfico aumenta en una unidad (el símbolo del gráfico y el símbolo en MarketBookAdd() no deben coincidir obligatoriamente). Al llamar a MarketBookRelease(), el contador de suscripciones del símbolo indicado en los límites del gráfico disminuye en una unidad. La transmisión de los eventos BookEvent de cualquier símbolo dentro del gráfico continúa hasta que el contador de suscripciones del símbolo dado no sea igual a cero. Por eso, es importante que cada programa MQL5 que contenga llamadas a MarketBookAdd(), al finalizar su funcionamiento deje de estar suscrita para recibir eventos de cada símbolo utilizado con la ayuda de MarketBookRelease(). Para ello, basta con que el número de llamadas a MarketBookAdd() y MarketBookRelease() de cada llamada sea par durante el tiempo de vida útil del programa MQL5. El uso de banderas o contadores propios de suscripciones dentro del programa permite trabajar de forma segura con los eventos de BookEvent y previene contra la desactivación de las suscripciones para obtener este evento en terceros programas dentro de un mismo gráfico.

© 2000-2025, MetaQuotes Ltd.

Los eventos BookEvent nunca se omiten y siempre se ponen en la cola, incluso si en este momento aún no ha finalizado el procesamiento del anterior evento BookEvent. En este caso, además, es necesario tener en cuenta que los eventos BookEvent se entregan por sí mismos y no portan en ellos el estado de la profundidad de mercado. Esto significa que la llamada a MarketBookGet() desde el manejador OnBookEvent() permite recibir el estado actual de la profundidad de mercado en el momento de la llamada, y no el estado de la profundidad de mercado que provocó el envío del evento BookEvent. Para recibir todos los estados únicos de la profundidad de mercado de forma garantizada, la función OnBookEvent() deberá ser lo más rápida posible.

Ejemplo

```
//+------------------------------------------------------------------+
//|                                          OnBookEvent_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com/es/articles/2635"
#property version   "1.00"
#property description "Ejemplo de medición de la velocidad de la actualización de la p
#property description "El código se ha tomado del artículo https://www.mql5.com/es/art
//--- parámetros de entrada
input ulong ExtCollectTime   =30;  // tiempo de simulación en segundos
input ulong ExtSkipFirstTicks=10;  // número de ticks omitidos al inicio
//--- bandera de inicio de suscripción para recibir los eventos de BookEvent
bool book_subscribed=false;
//--- matriz para recibir las solicitudes de la profundidad de mercado
MqlBookInfo  book[];
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mostramos el inicio
   Comment(StringFormat("Esperamos la llegada de los primeros %I64u ticks",ExtSkipFirs
   PrintFormat("Esperamos la llegada de los primeros %I64u ticks",ExtSkipFirstTicks);
//--- activamos la retransmisión de la profundidad de mercado
   if(MarketBookAdd(_Symbol))
     {
      book_subscribed=true;
      PrintFormat("%s: la función MarketBookAdd(%s) ha retornado true",__FUNCTION__,_S
     }
   else
      PrintFormat("%s: la función MarketBookAdd(%s) ha retornado false! GetLastError()
//--- la inicialización ha tenido éxito
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Deinitialize expert                                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- mostramos el código del motivo de la desinicialización
   Print(__FUNCTION__,": Código del motivo de la desinicialización = ",reason);
//--- cancelamos nuestra suscripción para recibir los eventos de la profundidad del me
   if(book_subscribed)
     {
      if(!MarketBookRelease(_Symbol))
         PrintFormat("%s: MarketBookRelease(%s) ha retornado false! GetLastError()=%d"
      else
         book_subscribed=false;
     }
//---
  }
//+------------------------------------------------------------------+
//| BookEvent function                                               |
//+------------------------------------------------------------------+
void OnBookEvent(const string &symbol)
  {
   static ulong starttime=0;              // hora de inicio de la simulación
   static ulong tickcounter=0;            // contador de actualizaciones de la profundi
//--- trabajamos con los eventos de la profundidad de mercado solo en el caso de que n
   if(!book_subscribed)
      return;
//--- calculamos las actualizaciones solo de nuestro símbolo
   if(symbol!=_Symbol)
      return;
//--- omitimos los primeros ticks para realizar la limpieza primaria de la cola y el c
   tickcounter++;
   if(tickcounter<ExtSkipFirstTicks)
      return;
//--- recordamos la hora de inicio
   if(tickcounter==ExtSkipFirstTicks)
      starttime=GetMicrosecondCount();
//--- solicitamos los datos de la profundidad de mercado
   MarketBookGet(symbol,book);
//--- ¿cuando debemos detenernos?
   ulong endtime=GetMicrosecondCount()-starttime;
   ulong ticks  =1+tickcounter-ExtSkipFirstTicks;
// ¿cuánto tiempo ha pasado en microsegundos desde el inicio de la simulación?
   if(endtime>ExtCollectTime*1000*1000)
     {
      PrintFormat("%I64u ticks for %.1f seconds: %.1f ticks/sec ",ticks,endtime/1000.0
      ExpertRemove();
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostrar los contadores en el campo del comentario
   if(endtime>0)
      Comment(StringFormat("%I64u ticks for %.1f seconds: %.1f ticks/sec ",ticks,endti
  }
```

Ver también

MarketBookAdd, MarketBookRelease, MarketBookGet, OnTrade, OnTradeTransaction, OnTick, funciones de procesamiento de eventos, Ejecución de programas, Eventos del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

OnChartEvent

Se llama en los indicadores y expertos al suceder el eventoChartEvent. La función ha sido diseñada para el procesamiento de los cambios del gráfico provocados por las acciones del usario o el funcionamiento de los programas MQL5.

```
void  OnChartEvent()
   const int       id,       // identificador de evento
   const long&     lparam,   // parámetro del evento del tipo long
   const double&   dparam,   // parámetro del evento del tipo double
   const string&   sparam    // parámetro del evento del tipo string
   );
```

Parámetros

id

[in] Identificador de evento de la enumeración ENUM_CHART_EVENT.

lparam

[in] Parámetro de evento del tipo long

dparam

[in] Parámetro de evento del tipo double

sparam

[in] Parámetro de evento del tipo string

Valor retornado

No hay valor retornado

Observación

Existen 11 tipos de eventos que pueden ser procesados con la ayuda de la función predeterminada OnChartEvent(). Para los eventos personalizados se ha previsto 65535 identificadores en el rango que va desde CHARTEVENT_CUSTOM hasta CHARTEVENT_CUSTOM_LAST, incluido. Para generar un evento personalizado es imprescindible usar la función EventChartCustom().

Breve descripción de los eventos de la lista ENUM_CHART_EVENT:

- CHARTEVENT_KEYDOWN — pulsar el teclado cuando la ventana del gráfico se encuentra en el foco;
- CHARTEVENT_MOUSE_MOVE — desplazar el ratón y pulsar los botones del ratón (si para el gráfico se ha establecido la propiedad CHART_EVENT_MOUSE_MOVE=true);
- CHARTEVENT_OBJECT_CREATE — crear un objeto gráfico (si para el gráfico se ha establecido la propiedad CHART_EVENT_OBJECT_CREATE=true);
- CHARTEVENT_OBJECT_CHANGE — cambiar las propiedades del objeto a través de la venta de diálogo de propiedades;
- CHARTEVENT_OBJECT_DELETE — eliminar objeto gráfico (si para el gráfico se ha establecido la propiedad CHART_EVENT_OBJECT_DELETE=true);
- CHARTEVENT_CLICK — clic del ratón en el gráfico;
- CHARTEVENT_OBJECT_CLICK — clic del ratón en un objeto gráfico que pertenezca al gráfico;
- CHARTEVENT_OBJECT_DRAG — desplazar el objeto gráfico con la ayuda del ratón;

© 2000-2025, MetaQuotes Ltd.

- CHARTEVENT_OBJECT_ENDEDIT — finalizar la edición del texto en el campo de edición del objeto gráfico Edit (OBJ_EDIT);
- CHARTEVENT_CHART_CHANGE — cambios del gráfico;
- CHARTEVENT_CUSTOM+n — identificador de evento personalizado, donde n se encuentra en el rango de 0 a 65535. CHARTEVENT_CUSTOM_LAST contiene el último identificador permitido del evento personalizado (CHARTEVENT_CUSTOM+65535).

Todos los programas MQL5 funcionan en flujos distintos al flujo principal de la aplicación. El flujo principal del terminal es responsable de procesar todos los mensajes de sistema de Windows, y como resultado de este procesamiento, a su vez genera mensajes de Windows para su aplicación. Por ejemplo, el desplazamiento del ratón en el gráfico (evento WM_MOUSE_MOVE) genera varios mensajes de sistema para el posterior dibujado de la ventana de la aplicación; asimismo, envía mensajes internos a los expertos e indicadores iniciados en este gráfico. En este caso, además, puede darse la situación de que el flujo de la aplicación aún no haya tenido tiempo de procesar el mensaje de sistema WM_PAINT (y por eso no haya dibujado aún el gráfico), y el experto o indicador ya hayan recibido el evento sobre el desplazamiento del cursor del ratón. Entonces, en esta situación, la propiedad del gráfico CHART_FIRST_VISIBLE_BAR se cambiará solo después de dibujar el gráfico.

Para cada tipo de evento, los parámetros de entrada de la función OnChartEvent() tienen valores determinados, que son necesarios para el procesamiento de este evento. En el recuadro se enumeran los eventos transmitidos a través de los parámetros.

Evento	Valor del parámetro id	Valor del parámetro lparam	Valor del parámetro dparam	Valor del parámetro sparam
Evento de pulsación del teclado	CHARTEVENT_KEYDOWN	código de la tecla pulsada	Número de pulsaciones de la tecla generadas mientras se mantenía en estado de pulsada	Valor de línea de la máscara de bits que describe el estado de los botones del teclado
Evento de ratón (si para el gráfico se ha establecido la propiedad CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	coordenada X	coordenada Y	Valor de línea de la máscara de bits que describe el estado de los botones del ratón
Evento de ruleta del ratón (si para el gráfico se ha establecido la propiedad CHART_EVENT_MOUSE_WHEEL=true)	CHARTEVENT_MOUSE_WHEEL	Banderas de los estados de las teclas del ratón, coordenadas X e Y del cursor. La descripción se ofrece en el ejemplo	Valor Delta del giro de la ruleta del ratón	—

© 2000-2025, MetaQuotes Ltd.

Evento	Valor del parámetro id	Valor del parámetro lparam	Valor del parámetro dparam	Valor del parámetro sparam
Evento de creación de un objeto gráfico (si para el gráfico se ha establecido la propiedad CHART_EVENT_OBJECT_CREATE =true)	CHARTEVENT_OBJECT_CREATE	—	—	Nombre del objeto gráfico creado
Evento de cambio de las propiedades del objeto a través de la ventana de diálogo de propiedades	CHARTEVENT_OBJECT_CHANGE	—	—	Nombre del objeto gráfico cambiado
Evento de eliminación de un objeto gráfico (si para el gráfico se ha establecido la propiedad CHART_EVENT_OBJECT_DELETE =true)	CHARTEVENT_OBJECT_DELETE	—	—	Nombre del objeto gráfico eliminado
Evento de clic del ratón en el gráfico	CHARTEVENT_CLICK	coordenada X	coordenada Y	—
Evento de clic del ratón en el objeto gráfico	CHARTEVENT_OBJECT_CLICK	coordenada X	coordenada Y	Nombre del objeto gráfico en el que ha tenido lugar el evento
Evento de desplazamiento del objeto gráfico con la ayuda del ratón	CHARTEVENT_OBJECT_DRAG	—	—	Nombre del objeto gráfico desplazado
Evento de finalización de la edición del texto	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nombre del objeto gráfico "Campo de

© 2000-2025, MetaQuotes Ltd.

Evento	Valor del parámetro id	Valor del parámetro lparam	Valor del parámetro dparam	Valor del parámetro sparam
en el campo de edición del objeto gráfico "Campo de edición"				edición" en el que se ha editado el texto
Evento de cambio de tamaño del gráfico o cambio de las propiedades del gráfico a través de la ventana de diálogo de propiedades	CHARTEVENT_CHART_CHANGE	—	—	—
Evento personalizado con el número N	CHARTEVENT_CUSTOM+N	Valor establecido por la función EventChartCustom()	Valor establecido por la función EventChartCustom()	Valor establecido por la función EventChartCustom()

Ejemplo de oyente de eventos del gráfico:

```
//+------------------------------------------------------------------+
//|                                          OnChartEvent_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Ejemplo de oyente de eventos del gráfico y generador de eventos
//--- identificadores de teclas de servicio
#define KEY_NUMPAD_5       12
#define KEY_LEFT           37
#define KEY_UP             38
#define KEY_RIGHT          39
#define KEY_DOWN           40
#define KEY_NUMLOCK_DOWN   98
#define KEY_NUMLOCK_LEFT  100
#define KEY_NUMLOCK_5     101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP    104
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- mostramos el valor de la constante CHARTEVENT_CUSTOM
   Print("CHARTEVENT_CUSTOM=",CHARTEVENT_CUSTOM);
//---
   Print("Iniciado el experto con el nombre ",MQLInfoString(MQL5_PROGRAM_NAME));
//--- estableciendo la bandera de obtención de eventos de creación de objetos del grá
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- estableciendo la bandera de obtención de eventos de eliminación de objetos del g
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- activando los mensajes sobre el movimiento de la ruleta del ratón
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- la declaración forzosa de las propiedades del gráfico garantiza la preparación p
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- contador de ticks para generar un evento personalizado
   static int tick_counter=0;
//--- acumularemos los ticks en este número
   int simple_number=113;
//---
   tick_counter++;
//--- enviamos el evento personalizado si el contador de ticks es múltiplo de simple_n
   if(tick_counter%simple_number==0)
     {
      //--- formamos el identificador del evento personalizado en un rango de 0 a 6553
      ushort custom_event_id=ushort(tick_counter%65535);
      //---  enviamos un evento personalizado con rellenado de parámetros
      EventChartCustom(ChartID(),custom_event_id,tick_counter,SymbolInfoDouble(Symbol
      //--- mostramos el registro log para estudiar y analizar los resultados del ejer
      Print(__FUNCTION__,": Se ha enviado el evento personalizado ID=",custom_event_id
     }
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- pulsando un botón en el teclado
```

© 2000-2025, MetaQuotes Ltd.

```
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch((int)lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("KEY_NUMLOCK_LEFT pulsado");   break;
         case KEY_LEFT:          Print("KEY_LEFT pulsado");           break;
         case KEY_NUMLOCK_UP:    Print("KEY_NUMLOCK_UP pulsado");     break;
         case KEY_UP:            Print("KEY_UP pulsado");             break;
         case KEY_NUMLOCK_RIGHT: Print("KEY_NUMLOCK_RIGHT pulsado");  break;
         case KEY_RIGHT:         Print("KEY_RIGHT pulsado");          break;
         case KEY_NUMLOCK_DOWN:  Print("KEY_NUMLOCK_DOWN pulsado");   break;
         case KEY_DOWN:          Print("KEY_DOWN pulsado");           break;
         case KEY_NUMPAD_5:      Print("KEY_NUMPAD_5 pulsado");       break;
         case KEY_NUMLOCK_5:     Print("KEY_NUMLOCK_5 pulsado");      break;
         default:                Print("Se ha pulsado una tecla no enumerada");
        }
     }
//--- pulsando el botón izquierdo del ratón en el gráfico
   if(id==CHARTEVENT_CLICK)
      Print("Coordenadas del clic del ratón en el gráfico: x = ",lparam,"  y = ",dpara
//--- pulsando el ratón en un objeto gráfico
   if(id==CHARTEVENT_OBJECT_CLICK)
      Print("Pulsando el botón del ratón en el objeto con el nombre '"+sparam+"'");
//--- objeto eliminado
   if(id==CHARTEVENT_OBJECT_DELETE)
      Print("Se ha eliminado el objeto con el nombre ",sparam);
//--- creando objeto
   if(id==CHARTEVENT_OBJECT_CREATE)
      Print("Se ha creado el objeto con el nombre ",sparam);
//--- objeto modificado
   if(id==CHARTEVENT_OBJECT_CHANGE)
      Print("Se ha modificado el objeto con el nombre ",sparam);
//--- se han desplazado o modificado las coordenadas de los puntos de anclaje
   if(id==CHARTEVENT_OBJECT_DRAG)
      Print("Cambiando los puntos de anclaje del objeto con el nombre ",sparam);
//--- se ha modificado el texto en el campo de edición del objeto gráfico Edit
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      Print("Se ha modificado el texto en el objeto Edit ",sparam,"  id=",id);
//--- eventos de desplazamiento del ratón
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- analizamos el estado de los botones y la ruleta del ratón para este evento
      int flg_keys = (int)(lparam>>32);          // bandera de estado de las teclas Ct
      int x_cursor = (int)(short)lparam;         // coordenada X en la que ha tenido l
      int y_cursor = (int)(short)(lparam>>16);   // coordenada Y en la que ha tenido l
      int delta    = (int)dparam;                // valor sumado del giro de la ruleta
      //--- procesamos la bandera
```

© 2000-2025, MetaQuotes Ltd.

```
      string str_keys="";
      if((flg_keys&0x0001)!=0)
         str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0)
         str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0)
         str_keys+="SHIFT ";
      if((flg_keys&0x0008)!=0)
         str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0)
         str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0)
         str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0)
         str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1)+"'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
     }
//--- cambio de tamaño del gráfico o cambio de las propiedades del gráfico a través de
   if(id==CHARTEVENT_CHART_CHANGE)
      Print("Cambio de tamaño o propiedades del gráfico");
//--- evento personalizado
   if(id>CHARTEVENT_CUSTOM)
      PrintFormat("Evento personalizado ID=%d, lparam=%d, dparam=%G, sparam=%s",id,lpa
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse left
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse right
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse middle
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse first X key
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse second X key
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift key
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control key
   return(res);
  }
```

Ver también

EventChartCustom, Tipos de eventos del gráfico, funciones de procesamiento de eventos, Ejecución de programas, Eventos del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

OnTester

Se llama en los expertos al suceder el evento Tester para el procesamiento de las acciones necesarias al finalizar la simulación.

```
double  OnTester(void);
```

Valor retornado

Valor del criterio de optimización personalizado para analizar los resultados de la simulación.

Observación

La función OnTester() puede ser usada solo en expertos en la simulación, y ha sido pensada en primer lugar para calcular un cierto valor usado como criterio "Custom max" al optimizar los parámetros de entrada.

Al realizar la optimización genética, la clasificación de los resultados dentro de los límites de una generación se realiza con carácter descendente. Esto significa que se considerarán los mejores resultados desde el punto de vista del criterio de optimización aquellos que tengan un mayor valor. Los mejores criterios desde el punto de vista de la clasificación, se ubican al final y como consecuencia son descartados y no se usan en la siguiente generación.

De esta forma, con la ayuda de la función OnTester() es posible no solo crear y guardar informes propios de los resultados de la simulación, sino también gestionar el transcurso de la optimización para buscar los mejores parámetros de la estrategia comercial.

Ejemplo de cálculo del criterio de optimización personalizado. La idea consiste en calcular la regresión lineal del gráfico de balance, y es descrita en el artículo Optimizamos la estrategia del gráfico de balance y comparamos los resultados con el criterio "Balance + max Sharpe Ratio"

```
//+------------------------------------------------------------------+
//|                                                OnTester_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Ejemplo de asesor con el manejador OnTester()"
#property description "En calidad de criterio de optimización personalizado "
#property description "se retorna el coeficiente de regresión lineal del gráfico de ba
#property description "dividido entre el error medio cuadrático de la desviación"
//--- incluimos la clase de operaciones comerciales
#include <Trade\Trade.mqh>
//--- parámetros de entrada del experto
input double Lots              = 0.1;      // Volumen
input int    Slippage          = 10;       // Deslizamiento permitido
input int    MovingPeriod      = 80;       // Periodo de la media móvil
input int    MovingShift       = 6;        // Desplazmiento de la media móvil
//--- variables globales
int    IndicatorHandle=0;  // manejador del indicador
```

© 2000-2025, MetaQuotes Ltd.

```
bool   IsHedging=false;    // bandera de la cuenta
CTrade trade;              // para realizar transacciones comerciales
//---
#define EA_MAGIC 18052018
//+------------------------------------------------------------------+
//| Comprobando las condiciones de apertura de posición              |
//+------------------------------------------------------------------+
void CheckForOpen(void)
  {
   MqlRates rt[2];
//--- comerciamos solo al inicio de una nueva barra
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," failed, no history");
      return;
     }
//--- volumen de ticks
   if(rt[1].tick_volume>1)
      return;
//--- obtenemos los valores de la media móvil
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer from iMA failed, no data");
      return;
     }
//--- comprobamos la presencia de la señal
   ENUM_ORDER_TYPE signal=WRONG_VALUE;
//--- la vela se abierto por encima, y se ha cerrado por debajo de la media móvil
   if(rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=ORDER_TYPE_BUY;    // señal de compra
   else // la vela se ha abierto por debajo, y se ha cerrado por encima de la media mó
     {
      if(rt[0].open<ma[0] && rt[0].close>ma[0])
         signal=ORDER_TYPE_SELL;// señal de venta
     }
//--- comprobaciones adicionales
   if(signal!=WRONG_VALUE)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
        {
         double price=SymbolInfoDouble(_Symbol,signal==ORDER_TYPE_SELL ? SYMBOL_BID:SY
         trade.PositionOpen(_Symbol,signal,Lots,price,0,0);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Comprobando las condiciones de cierre de posición                |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void CheckForClose(void)
  {
   MqlRates rt[2];
//--- comerciamos solo al inicio de una nueva barra
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," failed, no history");
      return;
     }
   if(rt[1].tick_volume>1)
      return;
//--- obtenemos los valores de la media móvil
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer from iMA failed, no data");
      return;
     }
//--- la posición ya había sido elegida anteriormente con la ayuda de PositionSelect()
   bool signal=false;
   long type=PositionGetInteger(POSITION_TYPE);
//--- la vela se ha abierto por encima, y se ha cerrado por debajo de la media móvil
   if(type==(long)POSITION_TYPE_SELL && rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=true;
//--- la vela se ha abierto por debajo, y se ha cerrado por encima de la media móvil:
   if(type==(long)POSITION_TYPE_BUY && rt[0].open<ma[0] && rt[0].close>ma[0])
      signal=true;
//--- comprobaciones adicionales
   if(signal)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
         trade.PositionClose(_Symbol,Slippage);
     }
//---
  }
//+------------------------------------------------------------------+
//| Elegimos la posición considerando el tipo de cuenta: Netting o Hedging
//+------------------------------------------------------------------+
bool SelectPosition()
  {
   bool res=false;
//--- eligiendo la posición para la cuenta de Hedging
   if(IsHedging)
     {
      uint total=PositionsTotal();
      for(uint i=0; i<total; i++)
        {
         string position_symbol=PositionGetSymbol(i);
```

© 2000-2025, MetaQuotes Ltd.

```
         if(_Symbol==position_symbol && EA_MAGIC==PositionGetInteger(POSITION_MAGIC))
           {
            res=true;
            break;
           }
        }
     }
//--- eligiendo la posición para la cuenta de Netting
   else
     {
      if(!PositionSelect(_Symbol))
         return(false);
      else
         return(PositionGetInteger(POSITION_MAGIC)==EA_MAGIC); //---проверка Magic nur
     }
//--- resultado de la ejecución
   return(res);
  }
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- establecemos el tipo de comercio: Netting o Hedging
   IsHedging=((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)==ACCOU
//--- inicializamos el objeto para controlar correctamente las posiciones
   trade.SetExpertMagicNumber(EA_MAGIC);
   trade.SetMarginMode();
   trade.SetTypeFillingBySymbol(Symbol());
   trade.SetDeviationInPoints(Slippage);
//--- creamos el indicador Moving Average
   IndicatorHandle=iMA(_Symbol,_Period,MovingPeriod,MovingShift,MODE_SMA,PRICE_CLOSE);
   if(IndicatorHandle==INVALID_HANDLE)
     {
      printf("Error al crear el indicador iMA");
      return(INIT_FAILED);
     }
//--- ok
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick(void)
  {
//--- si la posición ya está abierta, comprobamos la condición de cierre
   if(SelectPosition())
      CheckForClose();
// comprobamos la condición de cierre de la posición
```

© 2000-2025, MetaQuotes Ltd.

```
   CheckForOpen();
//---
  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- valor del criterio de optimización personalizado (cuanto mayor sea, mejor)
   double ret=0.0;
//--- obtenemos los resultados de las transacciones en una matriz
   double array[];
   double trades_volume;
   GetTradeResultsToArray(array,trades_volume);
   int trades=ArraySize(array);
//--- si hay menos de 10 transacciones, la simulación no habrá dado buenos resultados
   if(trades<10)
      return (0);
//--- resultado medio por transacción
   double average_pl=0;
   for(int i=0;i<ArraySize(array);i++)
      average_pl+=array[i];
   average_pl/=trades;
//--- mostramos un mensaje para el modo de simulación única
   if(MQLInfoInteger(MQL_TESTER) && !MQLInfoInteger(MQL_OPTIMIZATION))
      PrintFormat("%s: Transacciones=%d, Beneficio medio=%.2f",__FUNCTION__,trades,ave
//--- calculamos los coeficientes de regresión lineal para el gráfico de beneficio
   double a,b,std_error;
   double chart[];
   if(!CalculateLinearRegression(array,chart,a,b))
      return (0);
//--- calculamos el error de desviación del gráfico con respecto a la línea de regres
   if(!CalculateStdError(chart,a,b,std_error))
      return (0);
//--- calculamos la relación del beneficio de tendencia con respecto a la desviación r
   ret=(std_error == 0.0) ? a*trades : a*trades/std_error;
//--- retornamos el valor del criterio de optimización personalizado
   return(ret);
  }
//+------------------------------------------------------------------+
//| Recibir la matriz de beneficio/pérdidas de las transacciones     |
//+------------------------------------------------------------------+
bool GetTradeResultsToArray(double &pl_results[],double &volume)
  {
//--- solicitamos la historia comercial completa
   if(!HistorySelect(0,TimeCurrent()))
      return (false);
   uint total_deals=HistoryDealsTotal();
   volume=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- establecemos el tamaño inicial de la matriz con espacio de sobra, según el núme
   ArrayResize(pl_results,total_deals);
//--- contador de transacciones que registra el resultado comercial: beneficio o pérd
   int counter=0;
   ulong ticket_history_deal=0;
//--- iteramos por todas las transacciones
   for(uint i=0;i<total_deals;i++)
     {
      //--- elegimos una transacción
      if((ticket_history_deal=HistoryDealGetTicket(i))>0)
        {
         ENUM_DEAL_ENTRY deal_entry  =(ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket_hi
         long            deal_type   =HistoryDealGetInteger(ticket_history_deal,DEAL_T
         double          deal_profit =HistoryDealGetDouble(ticket_history_deal,DEAL_PR
         double          deal_volume =HistoryDealGetDouble(ticket_history_deal,DEAL_VO
         //--- nos interesan solo las operaciones comerciales
         if((deal_type!=DEAL_TYPE_BUY) && (deal_type!=DEAL_TYPE_SELL))
            continue;
         //--- solo las transacciones que registran beneficio/pérdidas
         if(deal_entry!=DEAL_ENTRY_IN)
           {
            //--- anotamos el resultado comercial en la matriz e incrementamos el cont
            pl_results[counter]=deal_profit;
            volume+=deal_volume;
            counter++;
           }
        }
     }
//--- establecemos el tamaño definitivo de la matriz
   ArrayResize(pl_results,counter);
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcula una regresión lineal del tipo y=a*x+b                    |
//+------------------------------------------------------------------+
bool CalculateLinearRegression(double  &change[],double &chartline[],
                               double  &a_coef,double  &b_coef)
  {
//--- comprobamos que los datos sean suficientes
   if(ArraySize(change)<3)
      return (false);
//--- creamos una matriz del gráfico con acumulación
   int N=ArraySize(change);
   ArrayResize(chartline,N);
   chartline[0]=change[0];
   for(int i=1;i<N;i++)
      chartline[i]=chartline[i-1]+change[i];
//--- ahora calculamos los coeficientes de regresión
   double x=0,y=0,x2=0,xy=0;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<N;i++)
     {
      x=x+i;
      y=y+chartline[i];
      xy=xy+i*chartline[i];
      x2=x2+i*i;
     }
   a_coef=(N*xy-x*y)/(N*x2-x*x);
   b_coef=(y-a_coef*x)/N;
//---
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calcula el error medio cuadrático de la desviación para las a y b establecidas
//+------------------------------------------------------------------+
bool  CalculateStdError(double  &data[],double  a_coef,double  b_coef,double &std_err)
  {
//--- suma de cuadrados del error
   double error=0;
   int N=ArraySize(data);
   if(N<=2)
      return (false);
   for(int i=0;i<N;i++)
      error+=MathPow(a_coef*i+b_coef-data[i],2);
   std_err=MathSqrt(error/(N-2));
//---
   return (true);
  }
```

Ver también

Simulación de estrategias comerciales, TesterHideIndicators, Trabajo con los resultados de la optimización, TesterStatistics, OnTesterInit, OnTesterDeinit, OnTesterPass, MQL_TESTER, MQL_OPTIMIZATION, FileOpen, FileWrite, FileLoad, FileSave

© 2000-2025, MetaQuotes Ltd.

OnTesterInit

Se llama en los expertos al suceder el evento TesterInit para el procesamiento de las acciones necesarias antes de comenzar la optimización en el simulador de estrategias. Existen dos variantes de la función.

La versión con retorno del resultado

```
int  OnTesterInit(void);
```

Valor retornado

Valor del tipo int, cero indica que el experto iniciado en el gráfico antes del comienzo de la optimización se ha inicializado con éxito.

Es prioritario el uso de la llamada de OnTesterInit() con retorno del resultado de ejecución, ya que este método permite no solo ejecutar la inicialización del programa, sino también retornar el código de error en el caso de que el programa finalice antes de tiempo. Retorna cualquier valor distinto a INIT_SUCCEEDED (0); indica que existe un error y la optimización no será iniciada.

La versión sin retorno de resultado se ha dejado solo por compatibilidad con los códigos antiguos. No se recomienda su uso

```
void  OnTesterInit(void);
```

Observación

El evento TesterInit se genera de forma automática antes de comenzar la optimización del experto en el simulador de estrategias. Según este evento, el experto que tenga el manejador OnTesterDeinit() y/o OnTesterPass() se cargará automáticamente en un gráfico del terminal aparte con el símbolo y periodo indicado en el simulador.

Este experto obtiene los eventos TesterInit, TesterDeinit y TesterPass, pero no obtiene los eventos Init, Deinit y NewTick. Por consiguiente, la lógica necesaria para el procesamiento de los resultados de cada pasada en el proceso de optimización se debe implementar en los manejadores OnTesterInit(), OnTesterDeinit() y OnTesterPass().

El resultado de cada pasada única en la optimización de la estrategia se puede transmitir a través de un frame del manejador OnTester() con ayuda de la función FrameAdd().

La función OnTesterInit() se ha pensado para inicializar el experto antes de comenzar la optimización para el posterior procesamiento de los resultados de optimización. Siempre se usa de forma conjunta con el manejador OnTesterDeinit().

A la ejecución de OnTesterInit() se dedica un tiempo limitado; una vez se supera el mismo, el funcionamiento del experto finalizará forzosamente, mientras que la propia optimización será cancelada. En este caso, además, el Diario de registro del simulador mostrará el siguiente mensaje:

```
TesterOnTesterInit works too long. Tester cannot be initialized.
```

El ejemplo se ha tomado de OnTick, el manejador OnTesterInit() ha sido añadido para establecer los parámetros de optimización:

```
//+------------------------------------------------------------------+
//|                                        OnTesterInit_Sample.mq5 |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Ejemplo de asesor con el manejador OnTesterInit()"
#property description "en el que se establecen los valores y límites "
#property description "de los parámetros de entrada para la optimización"

input double lots=0. 1;        // volumen en lotes
input double kATR=3;           // longitud de la vela de señal en ATR
input int    ATRperiod=20;     // periodo del indicador ATR
input int    holdbars=8;       // cuántas barras mantenemos la posición
input int    slippage=10;      // Deslizamiento permitido
input bool   revers=false;     // ¿viramos la señal?
input ulong  EXPERT_MAGIC=0;   // MagicNumber del experto
//--- para guardar el manejador del indicador ATR
int atr_handle;
//--- aquí vamos a guardar los últimos valores de ATR y del cuerpo de la vela
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//--- recordamos la hora de comienzo de la optimización
datetime optimization_start;
//--- para mostrar la duración en un gráfico tras finalizar la optimización
string report;
//+------------------------------------------------------------------+
//| TesterInit function                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- establecemos los valores de los parámetros de entrada para la optimización
   ParameterSetRange("lots",false,0.1,0,0,0);
   ParameterSetRange("kATR",true,3.0,1.0,0.3,7.0);
   ParameterSetRange("ATRperiod",true,10,15,1,30);
   ParameterSetRange("holdbars",true,5,3,1,15);
   ParameterSetRange("slippage",false,10,0,0,0);
   ParameterSetRange("revers",true,false,false,1,true);
   ParameterSetRange("EXPERT_MAGIC",false,123456,0,0,0);
   Print("Se han establecido los valores iniciales y los límites de los parámetros de
//--- recordamos el inicio de la optimización
   optimization_start=TimeLocal();
   report=StringFormat("%s: la optimización ha sido iniciada en %s",
                       __FUNCTION__,TimeToString(TimeLocal(),TIME_MINUTES|TIME_SECONDS
//--- mostramos los mensajes en el gráfico y en el diario de registro del terminal
   Print(report);
   Comment(report);
//---
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| TesterDeinit function                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- duración de la optimización
   string log_message=StringFormat("%s: la optimización ha durado %d segundos",
                                   __FUNCTION__,TimeLocal()-optimization_start);
   PrintFormat(log_message);
   report=report+"\r\n"+log_message;
   Comment(report);
  }
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- inicializamos las variables globales
   last_atr=0;
   last_body=0;
//--- establecemos el volumen correcto
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- creamos el manejador del indicador ATR
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: no se ha logrado crear iATR, código de error %d",__FUNCTION__,G
      return(INIT_FAILED);
     }
//--- el experto se ha inicializado correctamente
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- señal comercial
   static int signal=0; // +1 indica la señal de compra, -1 indica la señal de venta
//--- comprobamos y cerramos las posiciones antiguas, abiertas hace más de "holdbars"
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- comprobamos la aparición de una nueva barra
   if(isNewBar())
     {
      //--- comprobamos la presencia de una señal
      signal=CheckSignal();
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si tenemos abierta una posición de compensación, omitimos la señal hasta que la
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
      signal=0;
      return; // salimos del manejador de eventos NewTick y no entramos en el mercado
     }
//--- para la cuenta de cobertura, cada posición vive y se cierra por separado
   if(signal!=0)
     {
      //--- señal de compra
      if(signal>0)
        {
         PrintFormat("%s: ¡Hay señal de compra! Revers=%s",__FUNCTION__,string(revers)
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- señal de venta
      if(signal<0)
        {
         PrintFormat("%s: ¡Hay señal de venta! Revers=%s",__FUNCTION__,string(revers))
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- final de la función OnTick
  }
//+------------------------------------------------------------------+
//| Comprueba la presencia de una señal comercial                    |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0 indica la ausencia de señal
   int res=0;
//--- obtenemos el valor de ATR en la penúltima barra finalizada (el índice de la bar
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- obtenemos los datos de la última barra cerrada en una matriz del tipo MqlR
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- calculamos el tamaño del cuerpo de la barra en la última barra cerrada
         last_body=bar[0].close-bar[0].open;
         //--- si el cuerpo de la última barra (con índice 1) supera el anterior valo
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // para la vela ascendente el valor es positivo
        }
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("%s: ¡No se ha podido obtener la última barra! Error",__FUNCTION_
     }
   else
      PrintFormat("%s: ¡No se ha podido obtener el valor del indicador ATR! Error",__F
//--- si está activado el modo de comercio reverso
   res=revers?-res:res;  // si es necesario, viramos la señal (en lugar de 1, retornar
//--- retornamos el valor de la señal comercial
   return (res);
  }
//+------------------------------------------------------------------+
//|  Retorna true al aparecer una nueva barra                        |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // guardamos la hora de apertura de la barra actual
//--- obtenemos la hora de apertura de la barra cero
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- si la hora de apertura ha cambiado, significa que ha aparecido una nueva barra
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- es necesario o no mostrar en el log la información sobre la hora de apertu
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- mostramos el mensaje sobre la hora de apertura de una nueva barra
         PrintFormat("%s: new bar on %s %s opened at %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- obtenemos los datos sobre el último tick
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() failed, error = ",GetLastError());
         //--- mostramos la hora del útlimo tick con una precisión de hasta un miliseg
         PrintFormat("Last tick was at %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- tenemos una nueva barra
      return (true);
     }
//--- no hay barras nuevas
   return (false);
  }
//+------------------------------------------------------------------+
//| Comprar según el mercado con un volumen establecido              |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- compramos al precio de mercado
```

© 2000-2025, MetaQuotes Ltd.

```
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Vender según el mercado con un volumen establecido               |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- vendemos al precio de mercado
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Cierre de posiciones según el tiempo de mantenimiento en barras  |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // número de posiciones abiertas
//--- iterar todas las posiciones abiertas
   for(int i=total-1; i>=0; i--)
     {
      //--- parámetros de la posición
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      datetime position_open=(datetime)PositionGetInteger(POSITION_TIME);
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;

      //--- si la posición vive durante bastante tiempo, y además el MagicNumber y el
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
         double volume=PositionGetDouble(POSITION_VOLUME);
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // ponemos el texto en minúsculas para formatear co
         PrintFormat("Cerramos la posición #%I64u %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- estableciendo el tipo de orden y enviando la solicitud comercial
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Preparando y enviando la solicitud comercial                     |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulon
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- declarando e inicializando las estructuras
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- parámetros de la solicitud
   request.action   =TRADE_ACTION_DEAL;                     // tipo de operación comer
   request.position =pos_ticket;                            // ticket de la posición,
   request.symbol   =Symbol();                              // símbolo
   request.volume   =volume;                                // volumen
   request.type     =type;                                  // tipo de orden
   request.price    =price;                                 // precio de finalización
   request.deviation=slip;                                  // desviación permitida re
   request.magic    =magicnumber;                           // MagicNumber de la order
//--- envío de la solicitud
   if(!OrderSend(request,result))
     {
      //--- mostramos la información sobre el fallo
      PrintFormat("OrderSend %s %s %.2f at %.5f error %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- comunicamos que la operación ha tenido éxito
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Ver también

Simulación de estrategias comerciales, Trabajo con los resultados de la optimización, OnTesterDeinit, OnTesterPass, ParameterGetRange, ParameterSetRange

© 2000-2025, MetaQuotes Ltd.

OnTesterDeinit

Se llama en los expertos al suceder el evento TesterDeinit para la ejecución de las acciones necesarias al finalizar la optimización del experto.

```
void  OnTesterDeinit(void);
```

Valor retornado

No hay valor retornado

Observación

El evento TesterDeinit se genera automáticamente al finalizar la optimización del experto en el simulador de estrategias.

Un experto que tenga el manejador OnTesterDeinit() o OnTesterPass(), al iniciar la optimización se carga automáticamente en un gráfico aparte del terminal, con el símbolo y periodo indicados en el simulador. La función ha sido pensada para el procesamiento final de todos los resultados de la optimización.

Es necesario recordar que los frames de optimización enviados por los agentes de simulación con la ayuda de la función FrameAdd() pueden llegar en paquetes y se requerirá tiempo para su entrega. Por eso, no todos los frames, y por consiguiente todos los eventos TesterPass pueden llegar y ser procesados en OnTesterPass() hasta el final de la optimización. Así, para obtener con garantías todos los frames con retrasos en OnTesterDeinit(), será necesario ubicar un bloque de código que use la función FrameNext().

Ver también

Simulación de estrategias comerciales, Trabajo con los resultados de la optimización, TesterStatistics, OnTesterInit, OnTesterPass, ParameterGetRange, ParameterSetRange

© 2000-2025, MetaQuotes Ltd.

OnTesterPass

Se llama en los expertos al suceder el evento TesterPass para el procesamiento de un nuevo frame de datos durante la optimización del experto.

```
void  OnTesterPass(void);
```

Valor retornado

No hay valor retornado

Observación

El evento TesterPass se genera automáticamente al llegar el frame durante la optimización del experto en el simulador de estrategias.

Un experto que tenga el manejador OnTesterDeinit() o OnTesterPass(), al iniciar la optimización se carga automáticamente en un gráfico aparte del terminal, con el símbolo y periodo indicados en el simulador. La función está pensada para procesar los frames obtenidos de los agentes de simulación durante la optimización. El envío del frame con los resultados de la simulación se debe realizar desde el manejador OnTester() con la ayuda de la función FrameAdd().

Es necesario recordar que los frames de optimización enviados por los agentes de simulación con la ayuda de la función FrameAdd() pueden llegar en paquetes y se requerirá tiempo para su entrega. Por eso, no todos los frames, y por consiguiente todos los eventos TesterPass pueden llegar y ser procesados en OnTesterPass() hasta el final de la optimización. Así, para obtener con garantías todos los frames con retrasos en OnTesterDeinit(), será necesario ubicar un bloque de código que use la función FrameNext().

Después de finalizar la optimización de OnTesterDeinit(), es posible iterar de nuevo todos los frames recibidos usando las funciones FrameFirst()/FrameFilter y FrameNext().

Ver también

Simulación de estrategias comerciales, Trabajo con los resultados de la optimización, OnTesterInit, OnTesterDeinit, FrameFirst, FrameFilter, FrameNext, FrameInputs

© 2000-2025, MetaQuotes Ltd.

Obtención de información de mercado

Son las funciones que sirven para conseguir la información sobre el estado del mercado.

Función	Acción
SymbolsTotal	Devuelve la cantidad de símbolos disponibles (seleccionados en MarketWatch o todos)
SymbolExist	Comprueba la existencia del símbolo con el nombre indicado
SymbolName	Devuelve el nombre del símbolo especificado
SymbolSelect	Selecciona un símbolo en la ventana MarketWatch o remueve un símbolo de la ventana
SymbolIsSynchronized	Comprueba la sincronización de datos para el símbolo especificado en el terminal con los datos en el servidor comercial
SymbolInfoDouble	Devuelve valor double del símbolo especificado para la propiedad correspondiente
SymbolInfoInteger	Devuelve el valor del tipo entero (long, datetime, int o bool) del símbolo especificado para la propiedad correspondiente
SymbolInfoString	Devuelve el valor string del símbolo especificado para la propiedad correspondiente
SymbolInfoMarginRate	Devuelve los coeficientes del margen dependiendo del tipo y la dirección de la orden
SymbolInfoTick	Devuelve los precios actuales para un símbolo especificado en una variable del tipo MqlTick
SymbolInfoSessionQuote	Permite obtener fecha y hora de apertura y de cierre de la sesión de cotización especificada para el símbolo y el día de la semana especificados.
SymbolInfoSessionTrade	Permite obtener fecha y hora de apertura y de cierre de la sesión de compraventa especificada para el símbolo y el día de la semana especificados.
MarketBookAdd	Proporciona la apertura de la profundidad de mercado (Depth Market) para el símbolo indicado, además se encarga de la suscripción a las notificaciones acerca del cambio de esta profundidad de mercado
MarketBookRelease	Proporciona el cierre de la profundidad de mercado (Depth Market) para el símbolo indicado, además da de baja la suscripción a las notificaciones acerca del cambio de esta profundidad de mercado
MarketBookGet	Devuelve un array de estructuras del tipo MqlBookInfo que contiene datos de la profundidad de mercado del símbolo especificado

© 2000-2025, MetaQuotes Ltd.

SymbolsTotal

Devuelve la cantidad de símbolos disponibles (seleccionados en MarketWatch o todos).

```
int  SymbolsTotal(
   bool  selected      // true - sólo los símbolos en MarketWatch
   );
```

Parámetros

selected

[in] Modo de solicitud. Puede adquirir valores true o false.

Valor devuelto

Si el parámetro "selected" es igual a true, se devuelve la cantidad de símbolos seleccionados en MarketWatch. Si el valor es false, se devuelve la cantidad total de todos los símbolos.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el número de símbolos disponibles
   int total    = SymbolsTotal(false);       // todos los símbolos en el servidor
   int selected = SymbolsTotal(true);        // todos los símbolos añadidos a la venta

//--- mostramos los datos obtenidos en el diario de registro
   PrintFormat("Number of available symbols on the server: %d\n"+
               "Number of available symbols selected in MarketWatch: %d", total, selec
   /*
   resultado:
   Number of available symbols on the server: 140
   Number of available symbols selected in MarketWatch: 11
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolExist

Comprueba la existencia del símbolo con el nombre indicado.

```
bool  SymbolExist(
   const string  name,     // nombre del símbolo
   bool&   is_custom       // señal del símbolo personalizado
   );
```

Parámetros

name

[in] Nombre del símbolo.

is_custom

[out] Señal del símbolo personalizado, que se establece si la ejecución tiene éxito. Si el valor es igual a true, el símbolo encontrado es personalizado.

Valor retornado

Retorna false si el símbolo no ha sido encontrado entre los símbolos estándar, ni tampoco entre los símbolos personalizados.

Ejemplo:

```
#define SYMBOL_NAME "GBPUSDn"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos una bandera de símbolo personalizado y verificamos la presencia de ur
   bool custom = false;
   bool result = SymbolExist(SYMBOL_NAME, custom);

//--- declaramos el texto del mensaje, por defecto será sobre un símbolo no encontrad
   string text = StringFormat("The symbol '%s' was not found among either the standar

//--- Si se encuentra un símbolo, crearemos un texto de mensaje dependiendo de en qué
   if(result)
     {
      //--- si se trata de un símbolo estándar
      if(!custom)
         text = StringFormat("The '%s' symbol is available on the server.", SYMBOL_NAM
      //--- si se trata de un símbolo de usuario
      else
         text = StringFormat("The symbol '%s' was found in the list of custom symbols.
     }

//--- mostramos en el diario de registro un mensaje sobre el resultado de la comproba
   Print(text);
```

© 2000-2025, MetaQuotes Ltd.

```
   /*
   resultado del símbolo estándar 'GBPUSD':
   The 'GBPUSD' symbol is available on the server.

   resultado para el símbolo personalizado 'GBPUSDx':
   The symbol 'GBPUSDx' was found in the list of custom symbols.

   resultado para el símbolo ausente 'GBPUSDn':
   The symbol 'GBPUSDn' was not found among either the standard or custom symbols.
   */
  }
```

Mire también

SymbolsTotal, SymbolSelect, Símbolos personalizados

© 2000-2025, MetaQuotes Ltd.

SymbolName

Devuelve el nombre del símbolo especificado.

```
string  SymbolName(
   int   pos,          // número en la lista
   bool  selected      // true - sólo los símbolos en MarketWatch
   );
```

Parámetros

pos

[in] Número del símbolo según la orden.

selected

[in] Modo de solicitud. Si el valor es true, el símbolo se coge de la lista de símbolos seleccionados en MarketWatch. Si el valor es false, el símbolo se coge de la lista general.

Valor devuelto

Valor del tipo string con el nombre del símbolo.

Ejemplo:

```
#define SYMBOL_NAME "GBPHKD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- establecemos la bandera de resultado de la búsqueda de símbolos en el servidor
   bool found = false;

//--- hallamos el símbolo 'SYMBOL_NAME' en la lista de todos los símbolos del servidor
   int total = SymbolsTotal(false);
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el nombre del símbolo en la lista según el índice del ciclo
      string name = SymbolName(i, false);

      //--- si este es el símbolo deseado, imprimiremos su nombre y posición en la lis
      if(name == SYMBOL_NAME)
        {
         PrintFormat("The '%s' symbol was found in the list of server symbols at index
         found = true;
         break;
        }
     }

//--- si el símbolo no se encuentra en el servidor, informaremos sobre ello antes de
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!found)
      PrintFormat("The '%s' symbol was not found on the server.", SYMBOL_NAME);
}
```

© 2000-2025, MetaQuotes Ltd.

SymbolSelect

Elige un símbolo o la ventana MarketWatch o remueve un símbolo de la ventana.

```
bool  SymbolSelect(
   string  name,        // nombre del símbolo
   bool    select       // añadir o remover
   );
```

Parámetros

name

[in] Nombre del símbolo.

select

[in] Conmutador. Si el valor es false, este símbolo tiene que ser removido de la ventana MarketWatch, de lo contrario el símbolo tiene que ser seleccionado en la ventana MarketWatch. El símbolo no puede ser removido si hay gráficos abiertos con este símbolo, o hay posiciones abiertas para este símbolo.

Valor devuelto

En caso de fallo la función devuelve false.

Ejemplo:

```
#define SYMBOL_NAME "GBPHKD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- comprobamos la presencia del símbolo en las listas, si no, informaremos sobre el
   bool custom = false;
   if(!SymbolExist(SYMBOL_NAME, custom))
     {
      PrintFormat("'%s' symbol not found in the lists", SYMBOL_NAME);
      return;
     }

//--- añadimos el símbolo a la ventana "Observación de mercado"
   ResetLastError();
   if(!SymbolSelect(SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }
//--- si el símbolo se ha añadido con éxito a la lista, obtendremos su índice en la ve
   int index = SymbolIndex(SYMBOL_NAME);
   PrintFormat("The '%s' symbol has been added to the MarketWatch list. Symbol index i
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ahora eliminaremos el símbolo de la ventana "Observación de mercado"
   ResetLastError();
   if(!SymbolSelect(SYMBOL_NAME, false))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }
//--- si el símbolo se elimina con éxito de la lista, su índice en la ventana "Observa
   index = SymbolIndex(SYMBOL_NAME);
   PrintFormat("The '%s' symbol has been removed from the MarketWatch list. Symbol inc

   /*
   resultado:
   The 'GBPHKD' symbol has been added to the MarketWatch list. Symbol index in the lis
   The 'GBPHKD' symbol has been removed from the MarketWatch list. Symbol index in the
   */
  }
//+------------------------------------------------------------------+
//| Retorna el índice del símbolo en la lista de "Observación de mercado"   |
//+------------------------------------------------------------------+
int SymbolIndex(const string symbol)
  {
   int total = SymbolsTotal(true);
   for(int i=0; i<total; i++)
     {
      string name = SymbolName(i, true);
      if(name == symbol)
         return i;
     }
   return(WRONG_VALUE);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolIsSynchronized

Comprueba si los datos para el símbolo especificado en el terminal están sincronizados con los datos en el servidor comercial.

```
bool  SymbolIsSynchronized(
   string  name,        // nombre del símbolo
   );
```

Parámetros

name

[in] Nombre del símbolo.

Valor devuelto

Si los datos están sincronizados, devuelve true, de lo contrario devuelve false.

Ejemplo:

```
#define SYMBOL_NAME "EURUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos la bandera de sincronización de datos en el terminal usando el símbolo
   bool    sync = SymbolIsSynchronized(SYMBOL_NAME);

//--- creamos un mensaje dependiendo de la bandera de sincronización
   string text = StringFormat("The data on the '%s' symbol in the terminal is synchron
   if(!sync)
      text = StringFormat("The data for the '%s' symbol in the terminal is not synchro

//--- mostramos en el registro el resultado obtenido
   Print(text);

   /*
  resultado para los datos sincronizados:
   The data on the 'EURUSD' symbol in the terminal is synchronized with the data on th

  resultado de los datos aún no sincronizados:
   The data for the 'GBPHKD' symbol in the terminal is not synchronized with the data
   */
  }
```

Véase también

SymbolInfoInteger, Organización de acceso a los datos

© 2000-2025, MetaQuotes Ltd.

SymbolInfoDouble

Devuelve la propiedad correspondiente del símbolo especificado. Existen 2 variantes de la función.

1. Inmediatamente devuelve el valor de la propiedad.

```
double  SymbolInfoDouble(
   string                    name,        // símbolo
   ENUM_SYMBOL_INFO_DOUBLE   prop_id      // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso de éxito el valor de la propiedad se coloca en una variable receptora que se pasa por referencia por el último parámetro.

```
bool  SymbolInfoDouble(
   string                    name,        // símbolo
   ENUM_SYMBOL_INFO_DOUBLE   prop_id,     // identificador de la propiedad
   double&                   double_var   // aquí recibimos el valor de la propiedad
   );
```

Parámetros

name

[in] Nombre del símbolo.

prop_id

[in] Identificador de la propiedad del símbolo. El valor puede ser uno de los valores de la enumeración ENUM_SYMBOL_INFO_DOUBLE.

double_var

[out] Variable del tipo double que recibe el valor de la propiedad solicitada.

Valor devuelto

Valor del tipo double. En caso de la ejecución fallida, la información sobre el error se puede obtener a través de la función GetLastError():

- 5040 - parámetro string erróneo para especificar el nombre del símbolo,
- 4301 - símbolo desconocido (instrumento financiero),
- 4302 - símbolo no seleccionado en "Observación del Mercado" (no figura en la lista de símbolos disponibles),
- 4303 - identificador erróneo de la propiedad del símbolo.

Nota

Si la función se utiliza para recibir la información sobre el último tick, será mejor utilizar SymbolInfoTick(). Es muy posible que para este símbolo aún no haya habido ninguna cotización desde el momento de conexión del terminal a la cuenta de trading. En este caso el valor que se solicita va a ser indeterminado.

En la mayoría de los casos será suficiente utilizar la función SymbolInfoTick() que en una llamada permite conseguir los valores Ask, Bid, Last, Volume y el tiempo de llegada del último tick.

La función SymbolInfoMarginRate() ofrece información sobre la cantidad del margen cobrado dependiendo del tipo y la dirección de la orden.

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
void OnTick()
  {
//--- obtenemos el spread de las propiedades del símbolo
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d puntos\r\n",
                            spreadfloat?"flotante":"fijo",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- ahora nosotros mismos calculamos el spread
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"Spread calculado = "+(string)spread_points+" puntos";
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoInteger

Devuelve la propiedad correspondiente del símbolo especificado. Existen 2 variantes de la función.

1. Inmediatamente devuelve el valor de la propiedad.

```
long  SymbolInfoInteger(
   string                     name,      // símbolo
   ENUM_SYMBOL_INFO_INTEGER   prop_id    // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso de éxito el valor de la propiedad se coloca en una variable receptora que se pasa por referencia por el último parámetro.

```
bool  SymbolInfoInteger(
   string                     name,      // símbolo
   ENUM_SYMBOL_INFO_INTEGER   prop_id,   // identificador de la propiedad
   long&                      long_var   // aquí recibimos el valor de la propiedad
   );
```

Parámetros

name

[in] Nombre del símbolo.

prop_id

[in] Identificador de la propiedad del símbolo. Su valor puede ser uno de los valores de la enumeración ENUM_SYMBOL_INFO_INTEGER.

long_var

[out] Variable del tipo long que recibe el valor de la propiedad solicitada.

Valor devuelto

Valor del tipo long. En caso de la ejecución fallida, la información sobre el error se puede obtener a través de la función GetLastError():

- 5040 - parámetro string erróneo para especificar el nombre del símbolo,
- 4301 - símbolo desconocido (instrumento financiero),
- 4302 - símbolo no seleccionado en "Observación del Mercado" (no figura en la lista de símbolos disponibles),
- 4303 - identificador erróneo de la propiedad del símbolo.

Nota

Si la función se utiliza para recibir la información sobre el último tick, será mejor utilizar SymbolInfoTick(). Es muy posible que para este símbolo aún no haya habido ninguna cotización desde el momento de conexión del terminal a la cuenta de trading. En este caso el valor que se solicita va a ser indeterminado.

En la mayoría de los casos será suficiente utilizar la función SymbolInfoTick() que en una llamada permite conseguir los valores Ask, Bid, Last, Volume y el tiempo de llegada del último tick.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnTick()
  {
//--- obtenemos el spread de las propiedades del símbolo
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d puntos\r\n",
                            spreadfloat?"flotante":"fijo",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- ahora nosotros mismos calculamos el spread
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"Spread calculado = "+(string)spread_points+" puntos";
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoString

Devuelve la propiedad correspondiente del símbolo especificado. Existen 2 variantes de la función.

1. Inmediatamente devuelve el valor de la propiedad.

```
string  SymbolInfoString(
   string                    name,         // símbolo
   ENUM_SYMBOL_INFO_STRING   prop_id       // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso de éxito el valor de la propiedad se coloca en una variable receptora que se pasa por referencia por el último parámetro.

```
bool  SymbolInfoString(
   string                    name,         // símbolo
   ENUM_SYMBOL_INFO_STRING   prop_id,      // identificador de la propiedad
   string&                   string_var    // aquí recibimos el valor de la propiedad
   );
```

Parámetros

name

[in] Nombre del símbolo.

prop_id

[in] Identificador de la propiedad del símbolo. Su valor puede ser uno de los valores de la enumeración ENUM_SYMBOL_INFO_STRING.

string_var

[out] Variable del tipo string que recibe el valor de la propiedad solicitada.

Valor devuelto

Valor del tipo string. En caso de la ejecución fallida, la información sobre el error se puede obtener a través de la función GetLastError():

- 5040 - parámetro string erróneo para especificar el nombre del símbolo,
- 4301 - símbolo desconocido (instrumento financiero),
- 4302 - símbolo no seleccionado en "Observación del Mercado" (no figura en la lista de símbolos disponibles),
- 4303 - identificador erróneo de la propiedad del símbolo.

Nota

Si la función se utiliza para recibir la información sobre el último tick, será mejor utilizar SymbolInfoTick(). Es muy posible que para este símbolo aún no haya habido ninguna cotización desde el momento de conexión del terminal a la cuenta de trading. En este caso el valor que se solicita va a ser indeterminado.

En la mayoría de los casos será suficiente utilizar la función SymbolInfoTick() que en una llamada permite conseguir los valores Ask, Bid, Last, Volume y el tiempo de llegada del último tick.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#define SYMBOL_NAME "USDJPY"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos los datos de línea del símbolo SYMBOL_NAME
   string currency_base   = SymbolInfoString(SYMBOL_NAME, SYMBOL_CURRENCY_BASE);     //
   string currency_profit = SymbolInfoString(SYMBOL_NAME, SYMBOL_CURRENCY_PROFIT);   //
   string currency_margin = SymbolInfoString(SYMBOL_NAME, SYMBOL_CURRENCY_MARGIN);   //
   string symbol_descript = SymbolInfoString(SYMBOL_NAME, SYMBOL_DESCRIPTION);       //

//--- creamos un texto de mensaje con los datos obtenidos
   string text=StringFormat("Symbol %s:\n"+
                            "- Currency Base: %s\n"+
                            "- Currensy Profit: %s\n"+
                            "- Currency Margin: %s\n"+
                            "- Symbol Description: %s",
                            SYMBOL_NAME, currency_base,
                            currency_profit, currency_margin,
                            symbol_descript);

//--- mostramos los datos obtenidos en el diario de registro
   Print(text);
   /*
   Symbol USDJPY:
   - Currency Base: USD
   - Currensy Profit: JPY
   - Currency Margin: USD
   - Symbol Description: US Dollar vs Yen
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoMarginRate

Returns the margin rates depending on the order type and direction.

```
bool  SymbolInfoMarginRate(
   string             name,                    // symbol name
   ENUM_ORDER_TYPE    order_type,              // order type
   double&            initial_margin_rate,     // initial margin rate
   double&            maintenance_margin_rate  // maintenance margin rate
   );
```

Parameters

name

[in] Symbol name.

order_type

[in] Order type.

initial_margin_rate

[in] A double type variable for receiving an initial margin rate. Initial margin is a security deposit for 1 lot deal in the appropriate direction. Multiplying the rate by the initial margin, we receive the amount of funds to be reserved on the account when placing an order of the specified type.

maintenance_margin_rate

[out] A double type variable for receiving a maintenance margin rate. Maintenance margin is a minimum amount for maintaining an open position of 1 lot in the appropriate direction. Multiplying the rate by the maintenance margin, we receive the amount of funds to be reserved on the account after an order of the specified type is activated.

Return Value

Returns true if request for properties is successful, otherwise false.

Ejemplo:

```
#define SYMBOL_NAME Symbol()

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos las variables en las que se anotarán los coeficentes
   double initial_margin_rate = 0;      // coeficiente del cobro de margen inicial
   double maintenance_margin_rate = 0; // coeficiente del cobro de margen de mantenimi

//--- obtenemos y mostramos en el diario de registro los coeficientes del símbolo SYMB
   SymbolInfoMarginRatePrint(SYMBOL_NAME, ORDER_TYPE_BUY, initial_margin_rate, mainter

//--- obtenemos y mostramos en el diario de registro los coeficientes del símbolo SYMB
   SymbolInfoMarginRatePrint(SYMBOL_NAME, ORDER_TYPE_SELL, initial_margin_rate, mainte
```

© 2000-2025, MetaQuotes Ltd.

```
   /*
   resultado:
   Initial margin rate for symbol AFKS for the Buy market order: 0.225600
   Maintenance margin rate for symbol AFKS for the Buy market order: 0.112800
   Initial margin rate for symbol AFKS for the Sell market order: 0.254400
   Maintenance margin rate for symbol AFKS for the Sell market order: 0.127200
   */
  }
//+------------------------------------------------------------------+
//| Mostramos en el registro los coeficientes de cobro de margen     |
//+------------------------------------------------------------------+
void SymbolInfoMarginRatePrint(const string symbol, const ENUM_ORDER_TYPE order_type,
  {
//--- obtenemos los coeficientes del símbolo symbol para la orden de mercado order_typ
   ResetLastError();
   if(!SymbolInfoMarginRate(symbol, order_type, initial_margin_rate, maintenance_margi
     {
      Print("SymbolInfoMarginRate() failed. Error ", GetLastError());
      return;
     }
//--- imprimimos los valores obtenidos de los coeficientes
   string type=(order_type==ORDER_TYPE_BUY ? "Buy" : order_type==ORDER_TYPE_SELL ? "Se
   PrintFormat("Initial margin rate for symbol %s for the %s market order: %f\n"+
               "Maintenance margin rate for symbol %s for the %s market order: %f",
               SYMBOL_NAME, type, initial_margin_rate,
               SYMBOL_NAME, type, maintenance_margin_rate);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoTick

Devuelve precios corrientes para un símbolo especificado en una variable del tipo MqlTick.

```
bool  SymbolInfoTick(
   string    symbol,     // símbolo
   MqlTick&  tick        // referencia a una estructura
   );
```

Parámetros

symbol

[in] Nombre del símbolo.

tick

[out] Referencia a una estructura del tipo MqlTick, en la que serán colocados los precios corrientes y la hora de la última renovación de precios.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Ejemplo:

```
#define SYMBOL_NAME "EURUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos un array con una estructura tipo MqlTick de dimensionalidad 1
   MqlTick tick[1]={};

//--- obtenemos los precios más recientes para el símbolo SYMBOL_NAME en la estructura
   if(!SymbolInfoTick(SYMBOL_NAME, tick[0]))
     {
      Print("SymbolInfoTick() failed. Error ", GetLastError());
      return;
     }

//--- mostramos los datos obtenidos en el diario de registro
   PrintFormat("Latest price data for the '%s' symbol:", SYMBOL_NAME);
   ArrayPrint(tick);
   /*
   resultado:
   Latest price data for the 'EURUSD' symbol:
                    [time]    [bid]    [ask] [last] [volume]     [time_msc] [flags] [volu
   [0] 2024.05.17 23:58:54 1.08685 1.08695 0.0000         0 1715990334319       6
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoSessionQuote

Permite obtener fecha y hora de apertura y de cierre de la sesión de cotización especificada para el símbolo y el día de la semana especificados.

```
bool  SymbolInfoSessionQuote(
   string            name,                // nombre del símbolo
   ENUM_DAY_OF_WEEK  day_of_week,         // día de la semana
   uint              session_index,       // número de sesión
   datetime&         from,                // hora de apertura de la sesión
   datetime&         to                   // hora de cierre de la sesión
   );
```

Parámetros

name

[in] Nombre del símbolo.

ENUM_DAY_OF_WEEK

[in] Día de la semana, se coge el valor de la enumeración ENUM_DAY_OF_WEEK.

uint

[in] Número ordinal de la sesión para la que hay que recibir la hora y fecha de apertura y de cierre. La indexación de las sesiones se empieza desde 0.

from

[out] Hora de apertura de la sesión en segundos transcurridos desde las 00 horas y 00 minutos. En el valor obtenido hay que ignorar la fecha.

to

[out] Hora de cierre de la sesión en segundos transcurridos desde las 00 horas y 00 minutos. En el valor obtenido hay que ignorar la fecha.

Valor devuelto

Si se obtienen los datos respecto a la sesión, símbolo y el día de la semana especificado, entonces la función devuelve true. De lo contrario, se devuelve false.

Ejemplo:

```
#define SYMBOL_NAME    Symbol()
#define SESSION_INDEX 0

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprimimos el encabezado con el símbolo y el índice de sesión SESSION_INDEX y
//--- en un ciclo por los días de la semana de lunes a viernes, imprimimos en el regis
   PrintFormat("Symbol %s, Quote session %d:", SYMBOL_NAME, SESSION_INDEX);
   for(int i=MONDAY; i<SATURDAY; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      SymbolInfoSessionQuotePrint(SYMBOL_NAME, (ENUM_DAY_OF_WEEK)i, SESSION_INDEX);
   /*
   resultado:
   Symbol RU000A103661, Quote session 0:
   - Monday     06:45 - 00:00
   - Tuesday    06:45 - 00:00
   - Wednesday  06:45 - 00:00
   - Thursday   06:45 - 00:00
   - Friday     06:45 - 00:00
   */
  }
//+------------------------------------------------------------------+
//| Mostramos en el registro la hora de inicio y finalización de     |
//| la sesión de cotización para los símbolos y días indicados       |
//+------------------------------------------------------------------+
void SymbolInfoSessionQuotePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- declaramos las variables para registrar el inicio y el final de la sesión de cot
   datetime date_from;  // hora de inicio de la sesión
   datetime date_to;    // hora de finalización de la sesión

//--- obtenemos los datos de la sesión de cotización
   if(!SymbolInfoSessionQuote(symbol, day_of_week, session_index, date_from, date_to))
     {
      Print("SymbolInfoSessionQuote() failed. Error ", GetLastError());
      return;
     }

//--- creamos el nombre del día de la semana a partir de una constante de enumeración
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- mostramos en el registro los datos de la sesión de cotización
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

Véase también

Información sobre el instrumento, TimeToStruct, Estructura de fecha

© 2000-2025, MetaQuotes Ltd.

SymbolInfoSessionTrade

Permite obtener fecha y hora de apertura y de cierre de la sesión de compraventa especificada para el símbolo y el día de la semana especificados.

```
bool  SymbolInfoSessionTrade(
   string            name,                // nombre del símbolo
   ENUM_DAY_OF_WEEK  day_of_week,         // día de la semana
   uint              session_index,       // número de sesión
   datetime&         from,                // hora de apertura de la sesión
   datetime&         to                   // hora de cierre de la sesión
   );
```

Parámetros

name

[in] Nombre del símbolo.

ENUM_DAY_OF_WEEK

[in] Día de la semana, se coge el valor de la enumeración ENUM_DAY_OF_WEEK.

uint

[in] Número ordinal de la sesión para la que hay que recibir la hora y fecha de apertura y de cierre. La indexación de las sesiones se empieza desde 0.

from

[out] Hora de apertura de la sesión en segundos transcurridos desde las 00 horas y 00 minutos. En el valor obtenido hay que ignorar la fecha.

to

[out] Hora de cierre de la sesión en segundos transcurridos desde las 00 horas y 00 minutos. En el valor obtenido hay que ignorar la fecha.

Valor devuelto

Si se obtienen los datos respecto a la sesión, símbolo y el día de la semana, entonces la función devuelve true. De lo contrario, se devuelve false.

Ejemplo:

```
#define SYMBOL_NAME    Symbol()
#define SESSION_INDEX 0

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprimimos el encabezado con el símbolo y el índice de sesión SESSION_INDEX y
//--- en un ciclo por los días de la semana de lunes a viernes, imprimimos las horas
   PrintFormat("Symbol %s, Trade session %d:", SYMBOL_NAME, SESSION_INDEX);
   for(int i=MONDAY; i<SATURDAY; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      SymbolInfoSessionTradePrint(SYMBOL_NAME, (ENUM_DAY_OF_WEEK)i, SESSION_INDEX);
   /*
   resultado:
   Symbol GBPUSD, Trade session 0:
   - Monday     00:15 - 23:55
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
   */
  }
//+------------------------------------------------------------------+
//| Mostramos en el registro la hora de inicio y finalización de     |
//| sesión comercial para el símbolo y el día indicado               |
//+------------------------------------------------------------------+
void SymbolInfoSessionTradePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- declaramos las variables para registrar el inicio y el final de la sesión comerc
   datetime date_from;  // hora de inicio de la sesión
   datetime date_to;    // hora de finalización de la sesión

//--- obtenemos los datos de la sesión comercial del símbolo y el día de la semana
   if(!SymbolInfoSessionTrade(symbol, day_of_week, session_index, date_from, date_to))
     {
      Print("SymbolInfoSessionTrade() failed. Error ", GetLastError());
ç      return;
     }

//--- creamos el nombre del día de la semana a partir de una constante de enumeración
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- mostramos en el registro los datos de la sesión comercial indicada
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

Véase también

Información sobre el instrumento, TimeToStruct, Estructura de fecha

© 2000-2025, MetaQuotes Ltd.

MarketBookAdd

Proporciona la apertura de profundidad de mercado (Depth Market) para un símbolo indicado, además se encarga de la suscripción a las notificaciones acerca del cambio de esta profundidad de mercado.

```
bool  MarketBookAdd(
   string  symbol      // símbolo
   );
```

Parámetros

symbol

[in] Nombre del símbolo cuya profundidad de mercado va a ser usada en el Asesor Experto o script.

Valor devuelto

Devuelve valor true en caso de apertura con éxito, de lo contrario devuelve false.

Nota

Normalmente, esta función debe invocarse desde la función OnInit() o en el constructor de clase. Para procesar las notificaciones que llegan en el programa de Asesor Experto tiene que haber la función void OnBookEvent(string& symbol).

Ejemplo:

```
#define    SYMBOL_NAME    "GBPUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- abrimos la profundidad de mercado del símbolo SYMBOL_NAME
   if(!MarketBookAdd(SYMBOL_NAME))
     {
      PrintFormat("MarketBookAdd(%s) failed. Error ", SYMBOL_NAME, GetLastError());
      return;
     }

//--- mostramos un mensaje en el diario de registro sobre la apertura exitosa de la pr
   PrintFormat("The MarketBook for the '%s' symbol was successfully opened and a subsc

//--- esperamos 2 segundos
   Sleep(2000);

//--- al finalizar, nos desuscribimos de la profundidad de mercado abierta
   ResetLastError();
   if(MarketBookRelease(SYMBOL_NAME))
      PrintFormat("MarketBook for the '%s' symbol was successfully closed", SYMBOL_NAM
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Error %d occurred when closing MarketBook using the '%s' symbol", G

   /*
   resultado:
   The MarketBook for the 'GBPUSD' symbol was successfully opened and a subscription t
   MarketBook for the 'GBPUSD' symbol was successfully closed
   */
  }
```

Véase también

Estructura de profundidad de mercado, Estructuras y clases

© 2000-2025, MetaQuotes Ltd.

MarketBookRelease

Proporciona el cierre de profundidad de mercado (Depth Market) para un símbolo indicado, además da de baja la suscripción a las notificaciones acerca del cambio de esta profundidad de mercado.

```
bool  MarketBookRelease(
   string  symbol      // nombre del símbolo
   );
```

Parámetros

symbol

[in] Nombre del símbolo.

Valor devuelto

Devuelve valor true en caso de cierre con éxito, de lo contrario devuelve false.

Nota

Normalmente, esta función debe invocarse desde la función OnDeinit(), si la función correspondiente MarketBookAdd() ha sido invocada en la función OnInit(). O tiene que llamarse del destructor de clase, si la función correspondiente MarketBookAdd() se invoca en el constructor de esta clase.

Ejemplo:

```
#define SYMBOL_NAME "GBPUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- abrimos la profundidad de mercado del símbolo SYMBOL_NAME
   if(!MarketBookAdd(SYMBOL_NAME))
     {
      PrintFormat("MarketBookAdd(%s) failed. Error ", SYMBOL_NAME, GetLastError());
      return;
     }

//--- mostramos un mensaje en el diario de registro sobre la apertura exitosa de la pr
   PrintFormat("The MarketBook for the '%s' symbol was successfully opened and a subsc

//--- esperamos 2 segundos
   Sleep(2000);

//--- al finalizar el funcionamiento, nos desuscribiremos de la profundidad de mercado
//--- mostramos mensajes en el diario registro sobre la cancelación exitosa de la susc
   ResetLastError();
   if(MarketBookRelease(SYMBOL_NAME))
      PrintFormat("MarketBook for the '%s' symbol was successfully closed", SYMBOL_NAM
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Error %d occurred when closing MarketBook using the '%s' symbol", G

   /*
   resultado:
   The MarketBook for the 'GBPUSD' symbol was successfully opened and a subscription t
   MarketBook for the 'GBPUSD' symbol was successfully closed
   */
  }
```

Véase también

Estructura de profundidad de mercado, Estructuras y clases

© 2000-2025, MetaQuotes Ltd.

MarketBookGet

Devuelve un array de estructuras del tipo MqlBookInfo que contiene datos de la profundidad de mercado de un símbolo especificado.

```
bool  MarketBookGet(
   string         symbol,      // símbolo
   MqlBookInfo&   book[]       // referencia a un array
   );
```

Parámetros

symbol

[in] Nombre del símbolo.

book[]

[out] Referencia a un array del historial de profundidad de mercado. El array puede ser previamente distribuido para una cantidad suficiente de apuntes. Si un array dinámico no ha sido previamente distribuido en la memoria operativa,el terminal de cliente lo distribuirá por sí mismo.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

La profundidad de mercado debe ser abierta previamente por la función MarketBookAdd().

Ejemplo:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo para ",Symbol());
      for(int i=0;i<size;i++)
        {
         Print(i+":",priceArray[i].price
               +"    Volume = "+priceArray[i].volume,
               " type = ",priceArray[i].type);
        }
     }
   else
     {
      Print("No se ha podido obtener el contenido de la profundidad de mercado para el
     }
```

Véase también

Estructura de profundidad de mercado, Estructuras y clases

© 2000-2025, MetaQuotes Ltd.

Funciones del calendario económico

En este apartado se describen las funciones para trabajar con el Calendario Económico, disponible directamente en la plataforma MetaTrader. El calendario económico es una enciclopedia ya preparada que contiene las descripciones de los más importantes indicadores macroeconómicos, sus fechas de salida y su nivel de importancia. Los valores actuales de los indicadores macroeconímicos llegan a la plataforma MetaTrader justo tras su publicación, y se representan en un gráfico en forma de rótulos: esto permitirá monitorear visualmente los índices necesarios en función del país, la divisa y la importancia.

Todas las funciones para trabajar con el Calendario Económico utilizan la hora del servidor comercial (TimeTradeServer). Esto significa que la hora en la estructura MqlCalendarValue y los parámetros de entrada de hora en las funciones CalendarValueHistoryByEvent/CalendarValueHistory se establecen en el huso horario del servidor comercial, y no en la hora local del usuario.

Las funciones del Calendario Económico permiten analizar de forma automática los eventos entrantes según nuestros propios criterios de importancia, y también en función de los países/divisas requeridos.

Función	Acción
CalendarCountryById	Obtiene la descripción del país según su identificador
CalendarEventById	Obtiene la descripción del evento según su identificador
CalendarValueById	Obtiene la descripción del valor del evento según su identificador
CalendarCountries	Obtiene la matriz de las descripciones de los países disponibles en el Calendario
CalendarEventByCountry	Obtiene la matriz de las descripciones de todos los eventos disponibles en el Calendario, según el código del país indicado
CalendarEventByCurrency	Obtiene la matriz de las descripciones de todos los eventos disponibles en el Calendario, según la divisa indicada
CalendarValueHistoryByEvent	Obtiene la matriz de valores de todos los eventos en el intervalo temporal indicado según el identificador del evento
CalendarValueHistory	Obtiene la matriz de valores de todos los eventos en el intervalo temporal indicado con filtro de país y/o divisa
CalendarValueLastByEvent	Obtiene la matriz de valores de un evento según su ID desde el momento en que se creó la base de datos del Calendario con el change_id indicado
CalendarValueLast	Obtiene la matriz de valores de todos los eventos con filtrado por país y/o divisa desde el momento en que se creó la base de datos del Calendario con el change_id indicado

© 2000-2025, MetaQuotes Ltd.

CalendarCountryById

Obtiene la descripción del país según su identificador.

```
bool  CalendarCountryById(
   const long           country_id,     // identificador del país
   MqlCalendarCountry&  country         // variable para obtener la descripción del pa
   );
```

Parámetros

country_id

[in] Identificador del país según el estándar ISO 3166-1.

country

[out] Variable del tipo MqlCalendarCountry para obtener la descripción del país.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false. Para obtener información sobre el error, necesitamos llamar la función GetLastError(). Posibles errores:

- 4001 - ERR_INTERNAL_ERROR (error general del sistema de ejecución),
- 5402 - ERR_CALENDAR_NO_DATA (el país no ha sido encontrado),
- 5401 - ERR_CALENDAR_TIMEOUT (se ha superado el límite de solicitud por tiempo).

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos la lista de países del Calendario Económico
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- comprobamos el resultado
   if(count==0)
      PrintFormat("CalendarCountries() returned 0! Error %d",GetLastError());
//--- si tenemos dos o más países
   if(count>=2)
     {
      MqlCalendarCountry country;
      //--- ahora obtenemos la descripción del país según su identificador
      if(CalendarCountryById(countries[1].id, country))
        {
         //--- preparamos la descripción del país
         string descr="id = "+IntegerToString(country.id)+"\n";
         descr+=("name = " + country.name+"\n");
         descr+=("code = " + country.code+"\n");
         descr+=("currency = " + country.currency+"\n");
         descr+=("currency_symbol = " + country.currency_symbol+"\n");
```

© 2000-2025, MetaQuotes Ltd.

```
         descr+=("url_name = " + country.url_name);
         //---mostramos la descripción del país
         Print(descr);
        }
      else
         Print("CalendarCountryById() failed. Error ",GetLastError());
     }
//---
  }
/*
  Resultado:
   id = 999
   name = European Union
   code = EU
   currency = EUR
   currency_symbol = €
   url_name = european-union
*/
```

Ver también

CalendarCountries, CalendarEventByCountry

© 2000-2025, MetaQuotes Ltd.

CalendarEventById

Obtiene la descripción del evento según su identificador.

```
bool  CalendarEventById(
   ulong                event_id,      // identificador del evento
   MqlCalendarEvent&    event          // variable para obtener la descripción del ever
   );
```

Parámetros

event_id

[in] Identificador del evento.

event

[out] Variable del tipo MqlCalendarEvent para obtener la descripción del evento.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false. Para obtener información sobre el error, necesitamos llamar la función GetLastError(). Posibles errores:

- 4001 - ERR_INTERNAL_ERROR (error general del sistema de ejecución),
- 5402 - ERR_CALENDAR_NO_DATA (el país no ha sido encontrado),
- 5401 - ERR_CALENDAR_TIMEOUT (se ha superado el límite de solicitud por tiempo).

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- código del país para Alemania según el estándar ISO 3166-1 Alpha-2
   string germany_code="DE";
//--- obtenemos los eventos para Alemania
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(germany_code,events);
//--- mostramos los eventos de Alemania en el Registro
   if(events_count>0)
     {
      PrintFormat("Eventos para Alemania: %d",events_count);
      ArrayPrint(events);
     }
   else
     {
      PrintFormat("No se ha logrado obtener los eventos para el código de país %s, er
                  germany_code,GetLastError());
      //--- finalizando el script de manera anticipada
      return;
     }
//--- obtenemos la descripción del último evento de la matriz events[]
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlCalendarEvent event;
   ulong event_id=events[events_count-1].id;
   if(CalendarEventById(event_id,event))
     {
      MqlCalendarCountry country;
      CalendarCountryById(event.country_id,country);
      PrintFormat("Se ha obtenido la descripción del evento de event_id=%d",event_id);
      PrintFormat("País: %s (código del país = %d)",country.name,event.country_id);
      PrintFormat("Nombre del evento: %s",event.name);
      PrintFormat("Código del evento: %s",event.event_code);
      PrintFormat("Importancia del evento: %s",EnumToString((ENUM_CALENDAR_EVENT_IMPOR
      PrintFormat("Tipo de evento: %s",EnumToString((ENUM_CALENDAR_EVENT_TYPE)event.ty
      PrintFormat("Sector del evento: %s",EnumToString((ENUM_CALENDAR_EVENT_SECTOR)eve
      PrintFormat("Periodicidad del evento: %s",EnumToString((ENUM_CALENDAR_EVENT_FREQ
      PrintFormat("Modo de salida del evento: %s",EnumToString((ENUM_CALENDAR_EVENT_TI
      PrintFormat("Unidad de medición del valor: %s",EnumToString((ENUM_CALENDAR_EVENT
      PrintFormat("Número de dígitos decimales: %d",event.digits);
      PrintFormat("Multiplicador del valor: %s",EnumToString((ENUM_CALENDAR_EVENT_MULT
      PrintFormat("URL de la fuente: %s",event.source_url);
     }
   else
      PrintFormat("No se ha logrado obtener la descripción del evento para event_d=%s,
                  event_id,GetLastError());
  }
/*
  Resultado:
  Eventos para Alemania: 50
             [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importa
   [ 0] 276010001      1        6           2           0          276      1
   [ 1] 276010002      1        6           2           0          276      1
   [ 2] 276010003      1        4           2           0          276      1
   [ 3] 276010004      1        4           2           0          276      1
   ....
   [47] 276500001      1        8           2           0          276      0
   [48] 276500002      1        8           2           0          276      0
   [49] 276500003      1        8           2           0          276      0
  Obtenidas descripciones del evento de event_id=276500003
  País: Germany (код страны = 276)
  Nombre del evento: Markit Composite PMI
  Código del evento: markit-composite-pmi
   Importancia del evento: CALENDAR_IMPORTANCE_MODERATE
   Tipo de evento: CALENDAR_TYPE_INDICATOR
   Sector del evento: CALENDAR_SECTOR_BUSINESS
   Periodicidad del evento: CALENDAR_FREQUENCY_MONTH
   Modo de salida del evento: CALENDAR_TIMEMODE_DATETIME
   Unidad de medición del valor: CALENDAR_UNIT_NONE
  Número de dígitos decimales: 1
   Multiplicador del valor: CALENDAR_MULTIPLIER_NONE
   URL de la fuente: https://www.markiteconomics.com
```

© 2000-2025, MetaQuotes Ltd.

```
*/
```

Ver también

CalendarEventByCountry, CalendarEventByCurrency, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueById

Obtiene la descripción del valor del evento según su identificador.

```
bool  CalendarValueById(
   ulong                value_id,      // identificador del valor del evento
   MqlCalendarValue&    value          // variable para obtener el valor del evento
   );
```

Parámetros

value_id

[in] Identificador del valor del evento.

value

[out] Variable del tipo MqlCalendarValue para obtener el valor del evento. Vea el siguiente ejemplo de procesamiento de eventos del calendario.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false. Para obtener información sobre el error, necesitamos llamar la función GetLastError(). Posibles errores:

- 4001 - ERR_INTERNAL_ERROR (error general del sistema de ejecución),
- 5402 - ERR_CALENDAR_NO_DATA (el país no ha sido encontrado),
- 5401 - ERR_CALENDAR_TIMEOUT (se ha superado el límite de solicitud por tiempo).

Observación

Todas las funciones para trabajar con el Calendario Económico utilizan la hora del servidor comercial (TimeTradeServer). Esto significa que la hora en la estructura MqlCalendarValue y los parámetros de entrada de hora en las funciones CalendarValueHistoryByEvent/CalendarValueHistory se establecen en el huso horario del servidor comercial, y no en la hora local del usuario.

La estructura MqlCalendarValue proporciona un método para obtener e comprobar valores a partir de los campos actual_value, forecast_value, prev_value y revised_prev_value. Si el valor del campo no está definido, éste almacenará el valor LONG_MIN (-9223372036854775808).

Es necesario tener en cuenta que los valores de estos campos se almacenan multiplicados por un factor de un millón. Eso quiere decir que cuando las funciones CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent y CalendarValueLast reciben valores en MqlCalendarValue, es necesario comprobar que los valores LONG_MIN de esos campos son idénticos; y si el valor está definido en el campo, para obtener el valor, el valor del campo debe dividirse entre 1000 000 (un millón). Otra forma de obtener valores es verificar y obtener valores mediante funciones de la propia estructura MqlCalendarValue.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- código del país para Japón según el estándar ISO 3166-1 Alpha-2
```

© 2000-2025, MetaQuotes Ltd.

```
   string japan_code="JP";
//--- establecemos los límites del intervalo del que tomamos los eventos
   datetime date_from=D'01.01.2018';  // tomamos todos los eventos desde el año 2018
   datetime date_to=0;                // 0 indica todos los eventos conocidos, incluso
//--- obtenemos la matriz de valores de los eventos para Japón
   MqlCalendarValue values[];
   int values_count=CalendarValueHistory(values,date_from,date_to,japan_code);
//--- iteramos por los valores encontrados de los eventos
   if(values_count>0)
     {
      PrintFormat("Número de valores para los eventos de Japón: %d",values_count);
      //--- eliminamos todos los valores "vacíos" (actual_value==-9223372036854775808)
      for(int i=values_count-1;i>=0;i--)
        {
         if(values[i].actual_value==-9223372036854775808)
            ArrayRemove(values,i,1);
        }
      PrintFormat("Número de valores después de eliminar los vacíos: %d",ArraySize(val
     }
   else
     {
      PrintFormat("No se ha logrado obtener los eventos para el código de país %s, err
                  japan_code,GetLastError());
      //--- finalizando el script de manera anticipada
      return;
     }
//--- dejamos no más de 10 valores en la matriz values[]
   if(ArraySize(values)>10)
     {
      PrintFormat("Reducimos la lista de valores hasta 10 y los mostramos");
      ArrayRemove(values,0,ArraySize(values)-10);
     }
   ArrayPrint(values);

//--- ahora mostramos cómo obtener la descripción del valor del evento usando una valu
   for(int i=0;i<ArraySize(values);i++)
     {
      MqlCalendarValue value;
      CalendarValueById(values[i].id,value);
      PrintFormat("%d: value_id=%d value=%d impact=%s",
                  i,values[i].id,value.actual_value,EnumToString(ENUM_CALENDAR_EVENT_I
     }
//---
  }
/*
  Resultado:
  Número de valores para los eventos de Japón: 1734
  Número de valores después de eliminar los vacíos: 1017
  Reducimos la lista de valores hasta 10 y los mostramos
```

© 2000-2025, MetaQuotes Ltd.

```
        [id] [event_id]              [time]            [period] [revision] [actual_val
   [0] 56500  392030004 2019.03.28 23:30:00 2019.03.01 00:00:00          0          900
   [1] 56501  392030005 2019.03.28 23:30:00 2019.03.01 00:00:00          0          700
   [2] 56502  392030006 2019.03.28 23:30:00 2019.03.01 00:00:00          0         1100
   [3] 56544  392030007 2019.03.28 23:30:00 2019.02.01 00:00:00          0         2300
   [4] 56556  392050002 2019.03.28 23:30:00 2019.02.01 00:00:00          0         1630
   [5] 55887  392020003 2019.03.28 23:50:00 2019.02.01 00:00:00          0          400
   [6] 55888  392020004 2019.03.28 23:50:00 2019.02.01 00:00:00          0        -1800
   [7] 55889  392020002 2019.03.28 23:50:00 2019.02.01 00:00:00          0          200
   [8] 55948  392020006 2019.03.28 23:50:00 2019.02.01 00:00:00          1         1400
   [9] 55949  392020007 2019.03.28 23:50:00 2019.02.01 00:00:00          1        -1000
  Mostramos un resumen informativo de los valores value_id
   0: value_id=56500 value=900000 impact=CALENDAR_IMPACT_POSITIVE
   1: value_id=56501 value=700000 impact=CALENDAR_IMPACT_NA
   2: value_id=56502 value=1100000 impact=CALENDAR_IMPACT_POSITIVE
   3: value_id=56544 value=2300000 impact=CALENDAR_IMPACT_NEGATIVE
   4: value_id=56556 value=1630000 impact=CALENDAR_IMPACT_POSITIVE
   5: value_id=55887 value=400000 impact=CALENDAR_IMPACT_NEGATIVE
   6: value_id=55888 value=-1800000 impact=CALENDAR_IMPACT_POSITIVE
   7: value_id=55889 value=200000 impact=CALENDAR_IMPACT_NEGATIVE
   8: value_id=55948 value=1400000 impact=CALENDAR_IMPACT_POSITIVE
   9: value_id=55949 value=-1000000 impact=CALENDAR_IMPACT_NEGATIVE
*/
```

Ver también

CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent, CalendarValueLast

© 2000-2025, MetaQuotes Ltd.

CalendarCountries

Obtiene la matriz de las descripciones de los países disponibles en el Calendario.

```
int  CalendarCountries(
   MqlCalendarCountry&  countries[]         // matriz para obtener la lista de descrip
   );
```

Parámetros

countries[]

[out] Matriz del tipo MqlCalendarCountry para obtener las descripciones de todos los países del Calendario.

Valor retornado

Número de descripciones obtenidas. Para obtener información sobre el error, necesitamos llamar la función GetLastError(). Posibles errores:

- 4001 - ERR_INTERNAL_ERROR (error general del sistema de ejecución),
- 5401 - ERR_CALENDAR_TIMEOUT (se ha superado el límite de solicitud por tiempo),
- 5400 - ERR_CALENDAR_MORE_DATA (el tamaño de la matriz es insuficiente para obtener las descripciones de todos los países, por eso ha recibido solo aquellas que cabían).

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos la lista de países del Calendario Económico
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- mostramos la matriz en el Registro
   if(count>0)
      ArrayPrint(countries);
   else
      PrintFormat("CalendarCountries() returned 0! Error %d",GetLastError());
/*
  Resultado:
        [id]           [name] [code] [currency] [currency_symbol]       [url_name] [re
   [ 0]    0 "Worldwide"      "WW"   "ALL"      ""                 "worldwide"
   [ 1]  999 "European Union" "EU"   "EUR"      "€"                "european-union"
   [ 2]  840 "United States"  "US"   "USD"      "$"                "united-states"
   [ 3]  124 "Canada"         "CA"   "CAD"      "$"                "canada"
   [ 4]   36 "Australia"      "AU"   "AUD"      "$"                "australia"
   [ 5]  554 "New Zealand"    "NZ"   "NZD"      "$"                "new-zealand"
   [ 6]  392 "Japan"          "JP"   "JPY"      "Ŧ"                "japan"
   [ 7]  156 "China"          "CN"   "CNY"      "Ŧ"                "china"
   [ 8]  826 "United Kingdom" "GB"   "GBP"      "J"                "united-kingdom"
```

© 2000-2025, MetaQuotes Ltd.

```
   [ 9]  756 "Switzerland"    "CH"   "CHF"      "₣"              "switzerland"
   [10]  276 "Germany"        "DE"   "EUR"      "€"              "germany"
   [11]  250 "France"         "FR"   "EUR"      "€"              "france"
   [12]  380 "Italy"          "IT"   "EUR"      "€"              "italy"
   [13]  724 "Spain"          "ES"   "EUR"      "€"              "spain"
   [14]   76 "Brazil"         "BR"   "BRL"      "R$"             "brazil"
   [15]  410 "South Korea"    "KR"   "KRW"      "₩"              "south-korea"
*/
  }
```

Ver también

CalendarEventByCountry, CalendarCountryById

© 2000-2025, MetaQuotes Ltd.

CalendarEventByCountry

Obtiene la matriz de las descripciones de todos los eventos disponibles en el Calendario, según el código del país indicado.

```
int  CalendarEventByCountry(
   string               country_code,     // nombre en clave del país ISO 3166-1 alpha
   MqlCalendarEvent&    events[]          // variable para obtener la matriz de descri
   );
```

Parámetros

country_code

[in] Nombre clave del país según ISO 3166-1 alpha-2

events[]

[out] Matriz del tipo MqlCalendarEvent para obtener las descripciones de todos los eventos para el país indicado.

Valor retornado

Número de descripciones obtenidas. Para obtener información sobre el error, necesitamos llamar la función GetLastError(). Posibles errores:

- 4001 - ERR_INTERNAL_ERROR (error general del sistema de ejecución),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insuficiente para ejecutar la solicitud),
- 5401 - ERR_CALENDAR_TIMEOUT (se ha superado el límite de solicitud por tiempo),
- error de ejecución ArrayResize()

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- código del país de la Unión Europea según el estándar ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- obtenemos los eventos de la Unión Europea
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- mostramos los eventos de la Unión Europea en el Registro
   if(events_count>0)
     {
      PrintFormat("Eventos para la Unión Europea: %d",events_count);
      ArrayPrint(events);
     }
//---
  }
/*
  Resultado:
  Eventos para la Unión Europea: 56
```

© 2000-2025, MetaQuotes Ltd.

```
              [id] [type]  [country_id] [unit] [importance] [multiplier] [digits] [ever
   [ 0] 999010001      0           999      0            2            0        0  "ECB
   [ 1] 999010002      0           999      0            2            0        0  "ECB
   [ 2] 999010003      0           999      0            3            0        0  "ECB
   [ 3] 999010004      0           999      0            3            0        0  "ECB
   [ 4] 999010005      0           999      0            2            0        0  "ECB
   [ 5] 999010006      1           999      1            3            0        2  "ECB
   [ 6] 999010007      1           999      1            3            0        2  "ECB
   [ 7] 999010008      0           999      0            2            0        0  "ECB
   [ 8] 999010009      1           999      2            2            3        3  "ECB
   [ 9] 999010010      0           999      0            2            0        0  "ECB
   [10] 999010011      0           999      0            2            0        0  "ECB
   ...

*/
```

Ver también

CalendarCountries, CalendarCountryById

© 2000-2025, MetaQuotes Ltd.

CalendarEventByCurrency

Obtiene la matriz de las descripciones de todos los eventos disponibles en el Calendario, según la divisa indicada.

```
int  CalendarEventByCurrency(
   const string         currency,     // nombre en clave de la divisa del país
   MqlCalendarEvent&    events[]      // variable para obtener la matriz de descripci
   );
```

Parámetros

currency

[in] Nombre clave de la divisa del país.

events[]

[out] Matriz del tipo MqlCalendarEvent para obtener las descripciones de todos los eventos para la divisa indicada.

Valor retornado

Número de descripciones obtenidas. Para obtener información sobre el error, necesitamos llamar la función GetLastError(). Posibles errores:

- 4001 - ERR_INTERNAL_ERROR (error general del sistema de ejecución),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insuficiente para ejecutar la solicitud),
- 5401 - ERR_CALENDAR_TIMEOUT (se ha superado el límite de solicitud por tiempo),
- error de ejecución ArrayResize()

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos la matriz para obtener los eventos del Calendario Económico
      MqlCalendarEvent events[];
//--- obtenos los eventos para la divisa de la Unión Europea
      int count = CalendarEventByCurrency("EUR",events);
      Print("count = ", count);
//--- para el ejemplo, serán suficientes 10 eventos
      if(count>10)
         ArrayResize(events,10);
//--- mostramos los eventos en el Registro
      ArrayPrint(events);
  }
/*
  Resultado:
             [id] [type] [country_id] [unit] [importance]
   [0] 999010001      0          999      0            2 "https://www.ecb.europa.eu/ho
```

© 2000-2025, MetaQuotes Ltd.

```
   [1] 999010002      0           999      0            2 "https://www.ecb.europa.eu/ho
   [2] 999010003      0           999      0            3 "https://www.ecb.europa.eu/ho
   [3] 999010004      0           999      0            3 "https://www.ecb.europa.eu/ho
   [4] 999010005      0           999      0            2 "https://www.ecb.europa.eu/ho
   [5] 999010006      1           999      1            3 "https://www.ecb.europa.eu/ho
   [6] 999010007      1           999      1            3 "https://www.ecb.europa.eu/ho
   [7] 999010008      0           999      0            2 "https://www.ecb.europa.eu/ho
   [8] 999010009      1           999      2            2 "https://www.ecb.europa.eu/ho
   [9] 999010010      0           999      0            2 "https://www.ecb.europa.eu/ho
*/
```

Ver también

CalendarEventById, CalendarEventByCountry

© 2000-2025, MetaQuotes Ltd.

CalendarValueHistoryByEvent

Obtiene la matriz de valores de todos los eventos en el intervalo temporal indicado según el identificador del evento.

```
int  CalendarValueHistoryByEvent(
   ulong              event_id,          // identificador del evento
   MqlCalendarValue&  values[],          // matriz para obtener las descripciones de l
   datetime           datetime_from,     // límite izquierdo del intervalo temporal
   datetime           datetime_to=0      // límite derecho del intervalo temporal
   );
```

Parámetros

event_id

[in] Identificador del evento.

values[]

[out] Matriz del tipo MqlCalendarValue para obtener los valores de los eventos. Vea el siguiente ejemplo de procesamiento de eventos del calendario.

datetime_from

[in] Fecha inicial del intervalo temporal desde el cual se eligen los eventos según el identificador indicado, en este caso, además, *datetime_from* < *datetime_to*.

datetime_to=0

[in] Fecha final del intervalo temporal desde el cual se eligen los eventos según el identificador indicado. Si el parámetro *datetime_to* no ha sido indicado (o es igual a 0), se retornarán todos los valores de los eventos desde la fecha indicada *datetime_from* existentes en la base de datos del Calendario, incluidos los eventos y valores de eventos planeados para el futuro.

Valor retornado

En caso de éxito, retornará el número de valores disponibles en el array "values", en caso contrario -1.. Para obtener información sobre el error, necesitamos llamar la función GetLastError(). Posibles errores:

- 4001 - ERR_INTERNAL_ERROR (error general del sistema de ejecución),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insuficiente para ejecutar la solicitud),
- 5401 - ERR_CALENDAR_TIMEOUT (se ha superado el límite de solicitud por tiempo),
- 5400 - ERR_CALENDAR_MORE_DATA (el tamaño de la matriz es insuficiente para obtener las descripciones de todos los valores, por eso ha recibido solo aquellas que cabían),
- error de ejecución ArrayResize()

Si para el valor del evento no existe alguno de los campos de abajo

```
   struct MqlCalendarValue
     {
      ...
      long              actual_value;              // valor actual del evento
      long              prev_value;                // valor anterior del evento
      long              revised_prev_value;        // valor anterior revisado del e
      long              forecast_value;            // valor pronosticado del evento
```

© 2000-2025, MetaQuotes Ltd.

```
      ...
     };
```

entonces el valor del campo ausente se retornará como INT64_MIN (-9223372036854775808). Mire en el ejemplo de abajo el valor del campo *revised_prev_value*.

Observación

Todas las funciones para trabajar con el Calendario Económico utilizan la hora del servidor comercial (TimeTradeServer). Esto significa que la hora en la estructura MqlCalendarValue y los parámetros de entrada de hora en las funciones CalendarValueHistoryByEvent/CalendarValueHistory se establecen en el huso horario del servidor comercial, y no en la hora local del usuario.

La estructura MqlCalendarValue proporciona un método para obtener e comprobar valores a partir de los campos actual_value, forecast_value, prev_value y revised_prev_value. Si el valor del campo no está definido, éste almacenará el valor LONG_MIN (-9223372036854775808).

Es necesario tener en cuenta que los valores de estos campos se almacenan multiplicados por un factor de un millón. Eso quiere decir que cuando las funciones CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent y CalendarValueLast reciben valores en MqlCalendarValue, es necesario comprobar que los valores LONG_MIN de esos campos son idénticos; y si el valor está definido en el campo, para obtener el valor, el valor del campo debe dividirse entre 1000 000 (un millón). Otra forma de obtener valores es verificar y obtener valores mediante funciones de la propia estructura MqlCalendarValue.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- código del país de la Unión Europea según el estándar ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- obtenemos los eventos de la Unión Europea
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- mostramos los eventos de la Unión Europea en el Registro
   if(events_count>0)
     {
      PrintFormat("Eventos para la Unión Europea: %d",events_count);
      //--- reducimos la lista de eventos, para el estudio basta con dejar 10
      ArrayResize(events,10);
      ArrayPrint(events);
     }
//--- vemos que el evento "ECB Interest Rate Decision" tiene event_id=999010007
   ulong event_id=events[6].id;         // la id de este evento puede cambiar en el Cal
   string event_name=events[6].name;    // nombre del evento del Calendario
   PrintFormat("Obtenemos los valores para el evento event_name=%s event_id=%d",event_
//--- obtenemos todos los valores del evento "ECB Interest Rate Decision"
   MqlCalendarValue values[];
//--- establecemos los límites del intervalo del que tomamos los eventos
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime date_from=0;           // tomamos todos los eventos desde el comienzo de l
   datetime date_to=D'01.01.2016'; // tomamos los eventos no anteriores al año 2016
   if(CalendarValueHistoryByEvent(event_id,values,date_from,date_to))
     {
      PrintFormat("Se han obtenido los valores %s: %d",
                  event_name,ArraySize(values));
      //--- reducimos la lista de valores, para el estudio basta con dejar 10
      ArrayResize(values,10);
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Error! No se ha logrado obtener los valores para el evento event_id
      PrintFormat("Código de error: %d",GetLastError());
     }
  }
//---
/*
  Resultado:
  Eventos para la Unión Europea: 56
            [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importan
   [0] 999010001      0        5           0           0          999      0
   [1] 999010002      0        5           0           0          999      0
   [2] 999010003      0        5           0           0          999      0
   [3] 999010004      0        5           0           0          999      0
   [4] 999010005      0        5           0           0          999      0
   [5] 999010006      1        5           0           0          999      1
   [6] 999010007      1        5           0           0          999      1
   [7] 999010008      0        5           0           0          999      0
   [8] 999010009      1        5           0           0          999      2
   [9] 999010010      0        5           0           0          999      0
  Obtenemos los valores para el evento event_name=ECB Interest Rate Decision event_id=
  Se han obtenido los valores del evento ECB Interest Rate Decision: 102
       [id] [event_id]              [time]             [period] [revision] [actual_valu
   [0] 2776  999010007 2007.03.08 11:45:00 1970.01.01 00:00:00          0        3750(
   [1] 2777  999010007 2007.05.10 11:45:00 1970.01.01 00:00:00          0        3750(
   [2] 2778  999010007 2007.06.06 11:45:00 1970.01.01 00:00:00          0        4000(
   [3] 2779  999010007 2007.07.05 11:45:00 1970.01.01 00:00:00          0        4000(
   [4] 2780  999010007 2007.08.02 11:45:00 1970.01.01 00:00:00          0        4000(
   [5] 2781  999010007 2007.09.06 11:45:00 1970.01.01 00:00:00          0        4000(
   [6] 2782  999010007 2007.10.04 11:45:00 1970.01.01 00:00:00          0        4000(
   [7] 2783  999010007 2007.11.08 12:45:00 1970.01.01 00:00:00          0        4000(
   [8] 2784  999010007 2007.12.06 12:45:00 1970.01.01 00:00:00          0        4000(
   [9] 2785  999010007 2008.01.10 12:45:00 1970.01.01 00:00:00          0        4000(
*/
```

Ver también

© 2000-2025, MetaQuotes Ltd.

CalendarCountries, CalendarEventByCountry, CalendarValueHistory, CalendarEventById, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueHistory

Obtiene la matriz de valores de todos los eventos en el intervalo temporal indicado con filtro de país y/o divisa.

```
int  CalendarValueHistory(
   MqlCalendarValue&  values[],              // matriz para obtener las descripciones
   datetime           datetime_from,         // límite izquierdo del intervalo tempora
   datetime           datetime_to=0          // límite derecho del intervalo temporal
   const string       country_code=NULL,     // nombre clave del país ISO 3166-1 alpha
   const string       currency=NULL          // nombre clave de la divisa del país
   );
```

Parámetros

values[]

[out] Matriz del tipo MqlCalendarValue para obtener los valores de los eventos. Vea el siguiente ejemplo de procesamiento de eventos del calendario.

datetime_from

[in] Fecha inicial del intervalo temporal desde el cual se eligen los eventos según el identificador indicado, en este caso, además, *datetime_from* < *datetime_to*.

datetime_to=0

[in] Fecha final del intervalo temporal desde el cual se eligen los eventos según el identificador indicado. Si el parámetro *datetime_to* no ha sido indicado (o es igual a 0), se retornarán todos los valores de los eventos desde la fecha indicada *datetime_from* existentes en la base de datos del Calendario, incluidos los eventos y valores de eventos planeados para el futuro.

country_code=NULL

[in] Nombre clave del país según ISO 3166-1 alpha-2

currency=NULL

[in] Nombre clave de la divisa del país.

Valor retornado

En caso de éxito, retornará el número de valores disponibles en el array "values", en caso contrario -1.. Para obtener información sobre el error, necesitamos llamar la función GetLastError(). Posibles errores:

- 4001 - ERR_INTERNAL_ERROR (error general del sistema de ejecución),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insuficiente para ejecutar la solicitud),
- 5401 - ERR_CALENDAR_TIMEOUT (se ha superado el límite de solicitud por tiempo),
- 5400 - ERR_CALENDAR_MORE_DATA (el tamaño de la matriz es insuficiente para obtener las descripciones de todos los valores, por eso ha recibido solo aquellas que cabían),
- error de ejecución ArrayResize()

Observación

Todas las funciones para trabajar con el Calendario Económico utilizan la hora del servidor comercial (TimeTradeServer). Esto significa que la hora en la estructura MqlCalendarValue y los parámetros de

© 2000-2025, MetaQuotes Ltd.

entrada de hora en las funciones CalendarValueHistoryByEvent/CalendarValueHistory se establecen en el huso horario del servidor comercial, y no en la hora local del usuario.

Si se ha transmitido a la función la matriz *events[]* de longitud fija, y como resultado de la solicitud no ha habido espacio suficiente para guardar el resultado completo, se generará el error ERR_CALENDAR_MORE_DATA (5400).

Si no se ha indicado el parámetro *datetime_to* (=0), se retornarán todos los valores de los eventos desde la fecha indicada *datetime_from* existentes en la base de datos del Calendario, incluidos los eventos y valores de eventos planeados para el futuro.

Para los filtros *country_code* y *currency* los valores NULL y "" son equivalentes: designan la ausencia de filtro.

Para *country_code* es mejor usar el campo *code* de la estructura MqlCalendarCountry, por ejemplo, "US", "RU" o "EU".

Para *currency* es mejor usar el campo *currency* de la estructura MqlCalendarCountry, por ejemplo, "USD", "RUB" o "EUR".

Los filtros se usan mediante conjunción, es decir, a través de un 'Y' lógico se eligen los valores solo de aquellos eventos para los que se cumplen simultáneamente las dos condiciones: país y divisa

La estructura MqlCalendarValue proporciona un método para obtener e comprobar valores a partir de los campos actual_value, forecast_value, prev_value y revised_prev_value. Si el valor del campo no está definido, éste almacenará el valor LONG_MIN (-9223372036854775808).

Es necesario tener en cuenta que los valores de estos campos se almacenan multiplicados por un factor de un millón. Eso quiere decir que cuando las funciones CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent y CalendarValueLast reciben valores en MqlCalendarValue, es necesario comprobar que los valores LONG_MIN de esos campos son idénticos; y si el valor está definido en el campo, para obtener el valor, el valor del campo debe dividirse entre 1000 000 (un millón). Otra forma de obtener valores es verificar y obtener valores mediante funciones de la propia estructura MqlCalendarValue.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- código del país de la Unión Europea según el estándar ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- obtenemos todos los valores de los eventos de la Unión Europea
   MqlCalendarValue values[];
//--- establecemos los límites del intervalo del que tomamos los eventos
   datetime date_from=D'01.01.2018';  // tomamos todos los eventos desde el año 2018
   datetime date_to=0;                // 0 indica todos los eventos conocidos, incluso
//--- solicitamos la historia de un evento de la Unión Europea desde el año 2018
   if(CalendarValueHistory(values,date_from,date_to,EU_code))
     {
      PrintFormat("Se han obtenido los valores de los eventos de country_code=%s: %d",
```

© 2000-2025, MetaQuotes Ltd.

```
                  EU_code,ArraySize(values));
      //--- reducimos el tamaño de la matriz para la muestra en el Registro
      ArrayResize(values,10);
//--- mostramos los valores de los eventos en el Registro
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Error! No se ha logrado obtener el evento del país country_code=%s'
      PrintFormat("Código de error: %d",GetLastError());
     }
//---
  }
/*
  Resultado:
  Se han obtenido los valores de los eventos de country_code=EU: 1384
        [id] [event_id]           [time]               [period] [revision]    [actual_v
   [0] 54215  999500001 2018.01.02 09:00:00 2017.12.01 00:00:00          3        6060(
   [1] 54221  999500002 2018.01.04 09:00:00 2017.12.01 00:00:00          3        5660(
   [2] 54222  999500003 2018.01.04 09:00:00 2017.12.01 00:00:00          3        5810(
   [3] 45123  999030005 2018.01.05 10:00:00 2017.11.01 00:00:00          0          60(
   [4] 45124  999030006 2018.01.05 10:00:00 2017.11.01 00:00:00          0         280(
   [5] 45125  999030012 2018.01.05 10:00:00 2017.12.01 00:00:00          1          90(
   [6] 45126  999030013 2018.01.05 10:00:00 2017.12.01 00:00:00          1         140(
   [7] 54953  999520001 2018.01.05 20:30:00 2018.01.02 00:00:00          0      127900(
   [8] 22230  999040003 2018.01.08 10:00:00 2017.12.01 00:00:00          0         910(
   [9] 22231  999040004 2018.01.08 10:00:00 2017.12.01 00:00:00          0        1840(
*/
```

Ver también

CalendarCountries, CalendarEventByCountry, CalendarValueHistoryByEvent, CalendarEventById, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueLastByEvent

Obtiene la matriz de valores de un evento según su ID desde el momento en que se creó la base de datos del Calendario con el change_id indicado.

```
int  CalendarValueLastByEvent(
   ulong                event_id,      // identificador del evento
   ulong&               change_id,     // identificador del valor del evento
   MqlCalendarValue&    values[]       // matriz para obtener las descripciones de los
   );
```

Parámetros

event_id

[in] Identificador del evento.

change_id

[in][out] Identificador del cambio.

values[]

[out] Matriz del tipo MqlCalendarValue para obtener los valores del evento. Vea el siguiente ejemplo de procesamiento de eventos del calendario.

Valor retornado

Número de valores de evento obtenidos. Para obtener información sobre el error, necesitamos llamar la función GetLastError(). Posibles errores:

- 4001 - ERR_INTERNAL_ERROR (error general del sistema de ejecución),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insuficiente para ejecutar la solicitud),
- 5401 - ERR_CALENDAR_TIMEOUT (se ha superado el límite de solicitud por tiempo),
- 5400 - ERR_CALENDAR_MORE_DATA (el tamaño de la matriz es insuficiente para obtener las descripciones de todos los valores, por eso ha recibido solo aquellas que cabían),
- error de ejecución ArrayResize()

Observación

Todas las funciones para trabajar con el Calendario Económico utilizan la hora del servidor comercial (TimeTradeServer). Esto significa que la hora en la estructura MqlCalendarValue y los parámetros de entrada de hora en las funciones CalendarValueHistoryByEvent/CalendarValueHistory se establecen en el huso horario del servidor comercial, y no en la hora local del usuario.

Si se ha transmitido a la función la matriz *events[]* de longitud fija, y como resultado de la solicitud no ha habido espacio suficiente para guardar el resultado completo, se generará el error ERR_CALENDAR_MORE_DATA (5400).

Si se ha transmitido *change_id* = 0, la función retorna siempre cero, pero en este caso, además, al *change_id* se retorna el estado actual de la base de datos del Calendario.

La función retorna la matriz de valores para la noticia indicada y el nuevo *change_id* que se puede usar para las siguientes llamadas de esta función para obtener los nuevos valores de la noticia. Así, llamando esta función con el último *change_id* conocido, es posible obtener las actualizaciones de los valores para la noticia indicada.

© 2000-2025, MetaQuotes Ltd.

La estructura MqlCalendarValue proporciona un método para obtener e comprobar valores a partir de los campos actual_value, forecast_value, prev_value y revised_prev_value. Si el valor del campo no está definido, éste almacenará el valor LONG_MIN (-9223372036854775808).

Es necesario tener en cuenta que los valores de estos campos se almacenan multiplicados por un factor de un millón. Eso quiere decir que cuando las funciones CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent y CalendarValueLast reciben valores en MqlCalendarValue, es necesario comprobar que los valores LONG_MIN de esos campos son idénticos; y si el valor está definido en el campo, para obtener el valor, el valor del campo debe dividirse entre 1000 000 (un millón). Otra forma de obtener valores es verificar y obtener valores mediante funciones de la propia estructura MqlCalendarValue.

Ejemplo de un experto que escucha la salida del informe de Nonfarm payrolls:

```
#property description "Ejemplo de uso de la función CalendarValueLastByEvent"
#property description " para captar la salida del informe del evento Nonfarm Payrolls.
#property description "Para ello, es necesario obtener el identificador actual de camb
#property description " de la base de datos del Calendario. Y después obtener con este
#property description " solo los nuevos eventos a través de una solicitud en el tempor
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create timer
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destroy timer
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- identificador de cambio de la base de datos del Calendario
   static ulong calendar_change_id=0;
//--- señal del primer inicio
   static bool first=true;
//--- identificador del evento
   static ulong event_id=0;
//--- nombre del evento
   static string event_name=NULL;
//--- matriz de valores de los eventos
   MqlCalendarValue values[];
//--- inicializamos - obtenemos el calendar_change_id actual
   if(first)
     {
      MqlCalendarEvent events[];
      //--- código del país para los EE.UU. según el estándar ISO 3166-1 Alpha-2
      string USA_code="US";
      //--- obtenemos los eventos para los Estados Unidos
      int events_count=CalendarEventByCountry(USA_code,events);
      //--- posición del evento que necesitamos en la matriz events
      int event_pos=-1;
      //--- mostramos los eventos de los Estados Unidos en el Registro
      if(events_count>0)
        {
         PrintFormat("%s: Eventos para los Estados Unidos: %d",__FUNCTION__,events_cou
         for(int i=0;i<events_count;i++)
           {
            string event_name_low=events[i].name;
            //--- convertimos el nombre del evento en registro menor
            if(!StringToLower(event_name_low))
              {
               PrintFormat("StringToLower() ha retornado el error %d",GetLastError());
               //--- salimos de la función de forma anticipada
               return;
              }
            //--- buscando el evento "Nonfarm Payrolls"
            if(StringFind(event_name_low,"nonfarm payrolls")!=-1)
              {
               //--- evento localizado, guardamos su id
               event_id=events[i].id;
               //--- registramos su nombre para el evento "Nonfarm Payrolls"
               event_name=events[i].name;
               //--- guardamos la posición de los eventos en la matriz events[]
               event_pos=i;
               //--- en realidad, en el Calendario existen varios eventos que contiene
               PrintFormat("Evento \"Nonfarm Payrolls\" localizado: event_id=%d  event
               //--- mire todos los eventos tras comentar más abajo el operador break,
               break;
              }
           }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- reducimos la lista, eliminamos los eventos después del evento "Nonfarm
      ArrayRemove(events,event_pos+1);
      //--- para que el estudio sea más cómodo, dejamos los 9 eventos antes de "Nor
      ArrayRemove(events,0,event_pos-9);
      ArrayPrint(events);
     }
   else
     {
      PrintFormat("%s: CalendarEventByCountry(%s) ha retornado 0 eventos, código de
                  USA_code,__FUNCTION__,GetLastError());
      //--- el trabajo ha finalizado de forma errónea, vamos a intentarlo de nuevo
      return;
     }

   //--- obtenemos para el evento indicado el identificador de cambio de la base de
   if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
     {
      //--- este bloque de código no puede ejecutarse en el primer inicio, pero lo
      PrintFormat("%s: Se ha obtenido el identificador actual de la base de datos c
                  __FUNCTION__,calendar_change_id);
      //--- mostramos la bandera y salimos hasta el siguiente evento del temporizad
      first=false;
      return;
     }
   else
     {
      //--- los datos no han sido obtenidos (esto es normal para el primer inicio),
      int error_code=GetLastError();
      if(error_code==0)
        {
         PrintFormat("%s: Se ha obtenido el identificador actual de la base de datc
                     __FUNCTION__,calendar_change_id);
         //--- mostramos la bandera y salimos hasta el siguiente evento del tempori
         first=false;
         //--- ahora tenemos el valor calendar_change_id
         return;
        }
      else
        {
         //--- y esto es realmente un error
         PrintFormat("%s: No se ha logrado obtener los valores para el evento event
         PrintFormat("Código de error: %d",error_code);
         //--- el trabajo ha finalizado de forma errónea, vamos a intentarlo de nue
         return;
        }
     }
  }

//--- tenemos el último valor conocido del identificador de cambio del Calendario (cha
```

© 2000-2025, MetaQuotes Ltd.

```
   ulong old_change_id=calendar_change_id;
//--- comprobamos si ha aparecido un valor nuevo del evento "Nonfarm Payrolls"
   if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
     {
      PrintFormat("%s: Se han obtenido nuevos eventos para \"%s\": %d",
                  __FUNCTION__,event_name,ArraySize(values));
      //--- mostramos en el Registro la información de la matriz values
      ArrayPrint(values);
      //--- mostramos en el Registro los valores del identificador anterior y el actua
      PrintFormat("%s: Anterior change_id=%d, nuevo change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
/*
      escriba aquí su propio código para procesar la publicación de los datos de "Noni
      */
     }
//---
  }
/*
  Resultado:
   OnTimer: Eventos para los EE.UU.: 202
   Evento "Nonfarm Payrolls" localizado: event_id=840030016  event_name=Nonfarm Payrol
            [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importar
   [0] 840030007      1        4           2           0          840      1
   [1] 840030008      1        4           2           0          840      1
   [2] 840030009      1        4           2           0          840      0
   [3] 840030010      1        4           2           0          840      0
   [4] 840030011      1        4           2           0          840      1
   [5] 840030012      1        4           2           0          840      1
   [6] 840030013      1        4           2           0          840      1
   [7] 840030014      1        4           2           0          840      1
   [8] 840030015      1        3           2           0          840      1
   [9] 840030016      1        3           2           0          840      4
   OnTimer: Se ha obtenido el identificador actual de la base de datos del Calendario:

*/
```

Ver también

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueLast

Obtiene la matriz de valores de todos los eventos con filtrado por país y/o divisa desde el momento en que se creó la base de datos del Calendario con el change_id indicado.

```
int  CalendarValueLast(
   ulong&               change_id,              // identificador  del cambio
   MqlCalendarValue&    values[],               // matriz para obtener las descripcione
   const string         country_code=NULL,      // nombre clave del país ISO 3166-1 alp
   const string         currency=NULL           // nombre clave de la divisa del país
   );
```

Parámetros

change_id

[in][out] Identificador del cambio.

values[]

[out] Matriz del tipo MqlCalendarValue para obtener los valores del evento. Vea el siguiente ejemplo de procesamiento de eventos del calendario.

country_code=NULL

[in] Nombre clave del país según ISO 3166-1 alpha-2

currency=NULL

[in] Nombre clave de la divisa del país.

Valor retornado

Número de valores de evento obtenidos. Para obtener información sobre el error, necesitamos llamar la función GetLastError(). Posibles errores:

- 4001 - ERR_INTERNAL_ERROR (error general del sistema de ejecución),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insuficiente para ejecutar la solicitud),
- 5401 - ERR_CALENDAR_TIMEOUT (se ha superado el límite de solicitud por tiempo),
- 5400 - ERR_CALENDAR_MORE_DATA (el tamaño de la matriz es insuficiente para obtener las descripciones de todos los valores, por eso ha recibido solo aquellas que cabían),
- error de ejecución ArrayResize()

Observación

Todas las funciones para trabajar con el Calendario Económico utilizan la hora del servidor comercial (TimeTradeServer). Esto significa que la hora en la estructura MqlCalendarValue y los parámetros de entrada de hora en las funciones CalendarValueHistoryByEvent/CalendarValueHistory se establecen en el huso horario del servidor comercial, y no en la hora local del usuario.

Si se ha transmitido a la función la matriz *events[]* de longitud fija, y como resultado de la solicitud no ha habido espacio suficiente para guardar el resultado completo, se generará el error ERR_CALENDAR_MORE_DATA (5400).

Si se ha transmitido *change_id* = 0, la función retorna siempre cero, pero en este caso, además, al *change_id* se retorna el estado actual de la base de datos del Calendario.

© 2000-2025, MetaQuotes Ltd.

Para los filtros *country_code* y *currency* los valores NULL y "" son equivalentes: designan la ausencia de filtro.

Para *country_code* es mejor usar el campo *code* de la estructura MqlCalendarCountry, por ejemplo, "US", "RU" o "EU".

Para *currency* es mejor usar el campo *currency* de la estructura MqlCalendarCountry, por ejemplo, "USD", "RUB" o "EUR".

Los filtros se usan mediante conjunción, es decir, a través de un 'Y' lógico se eligen los valores solo de aquellos eventos para los que se cumplen simultáneamente las dos condiciones: país y divisa

La función retorna la matriz de valores para la noticia indicada y el nuevo *change_id* que se puede usar para las siguientes llamadas de esta función para obtener los nuevos valores de la noticia. Así, llamando esta función con el último *change_id* conocido, es posible obtener las actualizaciones de los valores para la noticia indicada.

La estructura MqlCalendarValue proporciona un método para obtener e comprobar valores a partir de los campos actual_value, forecast_value, prev_value y revised_prev_value. Si el valor del campo no está definido, éste almacenará el valor LONG_MIN (-9223372036854775808).

Es necesario tener en cuenta que los valores de estos campos se almacenan multiplicados por un factor de un millón. Eso quiere decir que cuando las funciones CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent y CalendarValueLast reciben valores en MqlCalendarValue, es necesario comprobar que los valores LONG_MIN de esos campos son idénticos; y si el valor está definido en el campo, para obtener el valor, el valor del campo debe dividirse entre 1000 000 (un millón). Otra forma de obtener valores es verificar y obtener valores mediante funciones de la propia estructura MqlCalendarValue.

Ejemplo de un experto que escucha la aparición de eventos del Calendario Económico:

```
#property description "Ejemplo de uso de la función CalendarValueLast"
#property description " para crear un escuchador de eventos del Calendario Económico."
#property description "Para ello, es necesario obtener el identificador actual de camb
#property description " de la base de datos del Calendario. Y después obtener con este
#property description " solo los nuevos eventos a través de una solicitud en el tempor
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create timer
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destroy timer
   EventKillTimer();
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- identificador de cambio de la base de datos del Calendario
   static ulong calendar_change_id=0;
//--- señal del primer inicio
   static bool first=true;
//--- matriz de valores de los eventos
   MqlCalendarValue values[];
//--- inicializamos - obtenemos el calendar_change_id actual
   if(first)
     {
      //--- obtenemos el identificador de cambio de la base de datos del Calendario
      if(CalendarValueLast(calendar_change_id,values)>0)
        {
         //--- este bloque de código no puede ejecutarse en el primer inicio, pero lo
         PrintFormat("%s: Se ha obtenido el identificador actual de la base de datos c
                     __FUNCTION__,calendar_change_id);
         //--- mostramos la bandera y salimos hasta el siguiente evento del temporizad
         first=false;
         return;
        }
      else
        {
         //--- los datos no han sido obtenidos (esto es normal para el primer inicio),
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: Se ha obtenido el identificador actual de la base de datc
                        __FUNCTION__,calendar_change_id);
            //--- mostramos la bandera y salimos hasta el siguiente evento del tempori
            first=false;
            //--- ahora tenemos el valor calendar_change_id
            return;
           }
         else
           {
            //--- y esto es realmente un error
```

© 2000-2025, MetaQuotes Ltd.

```
            PrintFormat("%s: No se ha podido obtener el evento en CalendarValueLast. C
                        __FUNCTION__,error_code);
            //--- el trabajo ha finalizado de forma errónea, vamos a intentar inicial
            return;
           }
        }
     }

//--- tenemos el último valor conocido del identificador de cambio del Calendario (cha
   ulong old_change_id=calendar_change_id;
//--- comprobamos si han aparecido nuevos eventos del Calendario
   if(CalendarValueLast(calendar_change_id,values)>0)
     {
      PrintFormat("%s: Se han obtenido nuevos eventos del Calendario: %d",
                  __FUNCTION__,ArraySize(values));
      //--- mostramos en el Registro la información de la matriz values
      ArrayPrint(values);
      //--- mostramos en el Registro los valores del identificador anterior y el actua
      PrintFormat("%s: Anterior change_id=%d, nuevo change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
      //--- mostramos los nuevos eventos en el Registro
      ArrayPrint(values);
      /*
     escriba aquí su propio código para procesar la aparición de eventos
      */
     }
//---
  }
/*
  Ejemplo de funcionamiento del escuchador:
  OnTimer: Se ha obtenido el identificador actual de la base de datos del Calendario:
  OnTimer: Se han obtenido los nuevos eventos para el Calendario: 1
        [id] [event_id]              [time]            [period] [revision] [actual_val
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -5077
  OnTimer: Anterior change_id=33281792, nuevo change_id=33282048
        [id] [event_id]              [time]            [period] [revision] [actual_val
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -5077
  OnTimer: Se han obtenido los nuevos eventos para el Calendario: 1
        [id] [event_id]              [time]            [period] [revision]       [actu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -922337203
  OnTimer: Anterior change_id=33282048, nuevo change_id=33282560
        [id] [event_id]              [time]            [period] [revision]       [actu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -922337203

*/
```

Ver también

© 2000-2025, MetaQuotes Ltd.

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

Acceso a las series temporales y a los datos de indicadores

Estas son las funciones para trabajar con las series temporales e indicadores. Una serie temporal se diferencia de una matriz usual en que la indexación de los elementos de una serie temporal se realiza del final de la matriz al principio (de los datos más recientes a los más antiguos). Se recomienda usar sólo los arrays dinámicos para copiar los valores de series temporales e indicadores, porque las funciones de copiado están diseñadas para asignar de una manera independiente el tamaño necesario de los arrays que reciben los valores.

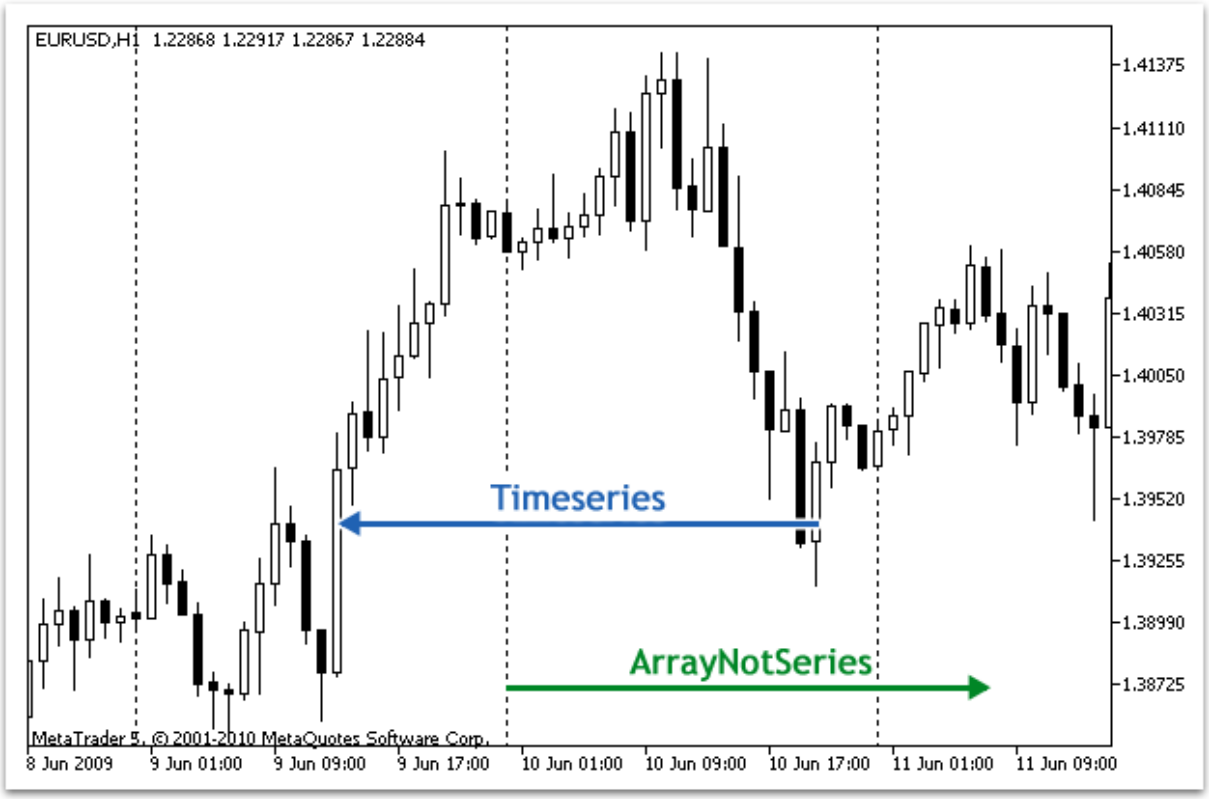
Hay una **importante excepción** de esta regla: si necesitamos copiar las series temporales y valores de indicadores con mucha frecuencia, por ejemplo, con cada nueva llamada a OnTick() en los Asesores Expertos o con cada nueva invocación de OnCalculate() en los indicadores, entonces sería mejor usar los arrays distribuidos estáticamente porque **las operaciones de asignación de memoria** para las matrices dinámicas **requieren su tiempo adicional** y eso tendrá su efecto durante los procesos de prueba y optimización.

Cuando usamos las funciones de acceso a las series temporales y a los valores de indicadores, hay que tener en cuenta la dirección de la indexación. Esto está descrito con detalles en la sección llamada Dirección de indexación en los arrays y series temporales.

El acceso a los datos de los indicadores y series temporales se realiza independientemente del hecho de disposición de estos datos solicitados (llamado el acceso asíncrono). Es sumamente importante para la calculación de los indicadores personalizados, por eso si los datos solicitados no están disponibles, las funciones como *Copy...()* inmediatamente devuelven el error. Sin embargo, si accedemos desde los Asesores Expertos o scripts, se hacen varios intentos de obtener los datos con una pequeña pausa que se necesita para proporcionar el tiempo necesario para cargar las series temporales que faltan o para calcular los valores de indicadores.

En el apartado Organización de acceso a los datos se explican minuciosamente los detalles de la obtención, almacenamiento y solicitud de los datos de precios en el terminal de cliente MetaTrader 5.

© 2000-2025, MetaQuotes Ltd.



Históricamente se ha constituido que el acceso a los datos en un array de precios se realiza desde el final de los datos. Físicamente los datos nuevos siempre se añaden al final del array, pero el índice de este array siempre es igual a cero. El índice 0 en una matriz-serie temporal significa los datos de la barra en curso, es decir, de la barra que corresponde al intervalo de tiempo no finalizado en dicho período de tiempo (timeframe).

Un período de tiempo (timeframe) es un plazo de tiempo durante el cual se forma una barra de precio. En total están predefinidos 21 períodos de tiempo estándares.

Función	Acción
SeriesInfoInteger	Devuelve la información sobre el estado de datos históricos
Bars	Devuelve la cantidad de barras en el historial por símbolo y período correspondientes
BarsCalculated	Devuelve la cantidad de datos calculados en el búfer de indicadores o -1 en caso del error (los datos aún no están calculados)
IndicatorCreate	Devuelve el manejador (handle) del indicador técnico especificado que ha sido creado a base del array de parámetros del tipo MqlParam
IndicatorParameters	Devuelve para el manejador especificado el número de los parámetros de entrada del indicador, así como los propios valores y el tipo de parámetros
IndicatorRelease	Elimina el manejador (handle) del indicador y libera la parte calculadora del indicador si nadie la está usando
CopyBuffer	Recibe en el array los datos de un búfer especificado desde un indicador especificado

© 2000-2025, MetaQuotes Ltd.

Función	Acción
CopyRates	Recibe en un array los datos históricos de la estructura Rates para un símbolo y período especificados
CopySeries	Obtiene en el conjunto especificado de arrays las series temporales sincronizadas de la estructura MqlRates para símbolo-periodo indicado en el número especificado.
CopyTime	Recibe en un array los datos históricos sobre el tiempo de apertura de barras para un símbolo y período especificados
CopyOpen	Recibe en un array los datos históricos sobre el precio de apertura de barras para un símbolo y período especificados
CopyHigh	Recibe en un array los datos históricos sobre el precio máximo de barras para un símbolo y período especificados
CopyLow	Recibe en un array los datos históricos sobre el precio mínimo de barras para un símbolo y período especificados
CopyClose	Recibe en un array los datos históricos sobre el precio de cierre de barras para un símbolo y período especificados
CopyTickVolume	Recibe en un array los datos históricos sobre volúmenes de tick para un símbolo y período especificados
CopyRealVolume	Recibe en un array los datos históricos sobre volúmenes comerciales para un símbolo y período especificados
CopySpread	Recibe en un array los datos históricos sobre los spreads para un símbolo y período especificados
CopyTicks	Recibe en un array los ticks acumulados por el terminal durante la sesión actual
iBars	Retorna el número de barras en la historia según el símbolo y el periodo correspondientes
iBarShift	Busca la barra según la hora y fecha. La función retorna el índice de la barra en el que entra la hora y fecha especificada
iClose	Retorna el valor del precio de cierre de la barra (indicada por el parámetro shift) del gráfico correspondiente
iHigh	Retorna el valor del precio mínimo de la barra (indicada por el parámetro shift) del gráfico correspondiente
iHighest	Retorna el índice del mayor valor encontrado (desplazamiento relativo a la barra actual) del gráfico correspondiente
iLow	Retorna el valor del precio mínimo de la barra (indicada por el parámetro shift) del gráfico correspondiente
iLowest	Retorna el índice del menor valor encontrado (desplazamiento relativo a la barra actual) del gráfico correspondiente

© 2000-2025, MetaQuotes Ltd.

Función	Acción
iOpen	Retorna el valor del precio de apertura de la barra (indicada por el parámetro shift) del gráfico correspondiente
iTime	Retorna el valor del tiempo de apertura de la barra (indicada por el parámetro shift) del gráfico correspondiente
iTickVolume	Retorna el valor del volumen de ticks de la barra (indicada por el parámetro shift) del gráfico correspondiente
iRealVolume	Retorna el valor del volumen real de la barra (indicada por el parámetro shift) del gráfico correspondiente
iVolume	Retorna el valor del volumen de ticks de la barra (indicada por el parámetro shift) del gráfico correspondiente
iSpread	Retorna el valor del spread de la barra (indicada por el parámetro shift) del gráfico correspondiente

A pesar de que mediante la función ArraySetAsSeries() para los arrays se pueda establecer un modo de acceso a los elementos igual que para las series temporales, hay que recordar que físicamente los elementos de un array siempre se almacenan en el mismo orden, sólo se cambia la dirección de la indexación. Para demostrarlo vamos a realizar el siguiente ejemplo:

```
   datetime TimeAsSeries[];
//--- Establecemos el acceso al array como a una serie temporal
   ArraySetAsSeries(TimeAsSeries,true);
   ResetLastError();
   int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
   if(copied<=0)
     {
      Print("No se ha podido copiar la hora de apertura de las últimas 10 barras");
      return;
     }
   Print("TimeCurrent = ",TimeCurrent());
   Print("ArraySize(Time) = ",ArraySize(TimeAsSeries));
   int size=ArraySize(TimeAsSeries);
   for(int i=0;i<size;i++)
     {
      Print("TimeAsSeries["+i+"] = ",TimeAsSeries[i]);
     }

   datetime ArrayNotSeries[];
   ArraySetAsSeries(ArrayNotSeries,false);
   ResetLastError();
   copied=CopyTime(NULL,0,0,10,ArrayNotSeries);
   if(copied<=0)
     {
      Print("No se ha podido copiar la hora de apertura de las últimas 10 barras");
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   size=ArraySize(ArrayNotSeries);
   for(int i=size-1;i>=0;i--)
     {
      Print("ArrayNotSeries["+i+"] = ",ArrayNotSeries[i]);
     }
```

Por lo tanto se mostrará algo parecido a lo siguiente:

```
TimeCurrent = 2009.06.11 14:16:23
ArraySize(Time) = 10
TimeAsSeries[0] = 2009.06.11 14:00:00
TimeAsSeries[1] = 2009.06.11 13:00:00
TimeAsSeries[2] = 2009.06.11 12:00:00
TimeAsSeries[3] = 2009.06.11 11:00:00
TimeAsSeries[4] = 2009.06.11 10:00:00
TimeAsSeries[5] = 2009.06.11 09:00:00
TimeAsSeries[6] = 2009.06.11 08:00:00
TimeAsSeries[7] = 2009.06.11 07:00:00
TimeAsSeries[8] = 2009.06.11 06:00:00
TimeAsSeries[9] = 2009.06.11 05:00:00

ArrayNotSeries[9] = 2009.06.11 14:00:00
ArrayNotSeries[8] = 2009.06.11 13:00:00
ArrayNotSeries[7] = 2009.06.11 12:00:00
ArrayNotSeries[6] = 2009.06.11 11:00:00
ArrayNotSeries[5] = 2009.06.11 10:00:00
ArrayNotSeries[4] = 2009.06.11 09:00:00
ArrayNotSeries[3] = 2009.06.11 08:00:00
ArrayNotSeries[2] = 2009.06.11 07:00:00
ArrayNotSeries[1] = 2009.06.11 06:00:00
ArrayNotSeries[0] = 2009.06.11 05:00:00
```

Como vemos, para el array TimeAsSeries con el aumento del índice el valor de tiempo con este índice se disminuye, es decir, nos movemos del presente al pasado. Para un array estándar ArrayNotSeries ocurre todo lo contrario; con el aumento del índice nos movemos del pasado al presente.

Véase también

ArrayIsDynamic, ArrayGetAsSeries, ArraySetAsSeries, ArrayIsSeries

© 2000-2025, MetaQuotes Ltd.

Dirección de indexación en los arrays y series temporales

Por defecto todas los arrays y búfers de indicadores tienen la dirección de indexación de izquierda a derecha. El índice del primer elemento siempre es igual a cero. De esta manera, el primer elemento de un array o búfer de indicador con el índice 0 por defecto se encuentra en el extremo izquierdo y el último elemento se encuentra en el extremo derecho.

Un búfer de indicador es un array dinámico del tipo double, cuyo tamaño es gestionado por el terminal de cliente para que éste siempre corresponda a la cantidad de barras sobre las cuales se calcula el indicador. Un array dinámico habitual del tipo double se asigna como un búfer de indicador a través de la función SetIndexBuffer(). Para los buffers de indicadores no hace falta establecer el tamaño mediante la función ArrayResize(), esto será hecho por el sistema de ejecución del mismo terminal.

Las series temporales son arrays con la indexación inversa. Es decir, el primer elemento de una serie temporal se encuentra en el extremo derecho y el último se encuentra en el izquierdo. Las series temporales están destinadas para almacenar los datos históricos de precios de los instrumentos financieros y contienen obligatoriamente la información sobre la hora, entonces se puede decir que en una serie temporal los datos más recientes se encuentran en el extremo derecho y los más antiguos en el extremo izquierdo.

Por tanto, en una serie temporal el elemento con el índice cero contiene la información sobre la última cotización de un instrumento. Si una serie temporal representa datos del período de tiempo diario, en la posición cero se contienen datos sobre el día en curso no finalizado, y en la posición con el índice uno se almacenan los datos del día de ayer.

Cambio de dirección de indexación

La función ArraySetAsSeries() permite cambiar el modo de acceso a los elementos de un array dinámico, pero el orden físico de almacenamiento de datos en la memoria del ordenador no sufre cambio alguno. Esta función simplemente cambia el modo de direccionamiento hacia los elementos de un array, por eso cuando copiamos un array dentro del otro a través de la función ArrayCopy(), el contenido del array-receptor no va a depender de la dirección de indexación en el array fuente.

No se puede cambiar la dirección de indexación para los arrays distribuidos estáticamente. Incluso si un array ha sido pasado como un parámetro a una función, dentro de esta función los intentos de cambiar la dirección de indexación no tendrán efecto alguno.

Para los búfers de indicadores, igual que para los arrays habituales, también se permite establecer la dirección de indexación al revés, como en las series temporales. Es decir, en este caso la referencia hacia la posición cero en el búfer de indicador va a significar la referencia hacia el último valor en el búfer de indicador correspondiente, y esto va a corresponder al valor del indicador en la última barra. Pero como ya se ha mencionado antes, la ubicación física de datos en el búfer de indicador se queda sin cambiar.

Recepción de datos de precios en los indicadores

Cada indicador personalizado obligatoriamente ha de contener la función OnCalculate(). A esta función se le pasan los datos de precios necesarios para calcular los valores en los búfers de indicadores. A través de la función ArrayGetAsSeries() se puede averiguar la dirección de indexación en estos arrays pasados.

© 2000-2025, MetaQuotes Ltd.

Los arrays pasados a la función reflejan los datos de precios, es decir, estos arrays tienen los indicios de las series temporales y la función ArrayIsSeries() devolverá true a la hora de comprobar estos arrays. Pero a pesar de eso, en cualquier caso hay que comprobar la dirección de indexación sólo a través de la función ArrayGetAsSeries().

Para no depender de los valores por defecto, hay que llamar incondicionalmente a la función ArraySetAsSeries() para los arrays con los que se prevé trabajar y establecer la dirección de indexación necesaria.

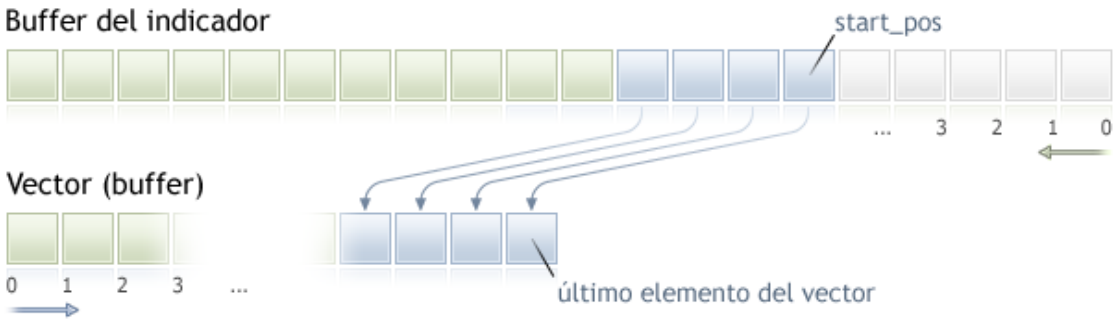
Recepción de datos de precios y valores de indicadores

Por defecto en los Asesores Expertos, indicadores y scripts todos los arrays tienen la dirección de indexación de izquierda a derecha. En caso de necesidad, en cualquier programa mql5 se puede solicitar los valores de las series temporales referente a cualquier símbolo y período de tiempo, además, los valores de indicadores calculados sobre cualquier símbolo y período de tiempo.

Para conseguir estos datos se usan las funciones Copy...():

- CopyBuffer - copia los valores de un búfer de indicador a un array del tipo double;
- CopyRates - copia el historial de precios a una matriz de estructuras MqlRates;
- CopyTime - copia los valores Time a un array del tipo datetime;
- CopyOpen - copia los valores Open a un array del tipo double;
- CopyHigh - copia los valores High a un array del tipo double;
- CopyLow - copia los valores Low a un array del tipo double;
- CopyClose - copia los valores Close a un array del tipo double;
- CopyTickVolume - copia los volúmenes de tick a un array del tipo long;
- CopyRealVolume - copia los volúmenes bursátiles a un array del tipo long;
- CopySpread - copia el historial de spreads a un array del tipo int;

Todas estas funciones trabajan de una manera igual, y por eso será suficiente estudiar el mecanismo de obtención de datos en el ejemplo de CopyBuffer(). Se supone que todos los datos solicitados tienen la dirección de indexación igual que en las series temporales, y en la posición con el índice 0 (cero) se almacenan los datos de la barra actual no finalizada. Para conseguir acceder a estos datos necesitamos copiar el volumen de datos que hace falta a un array-receptor, por ejemplo al array *buffer*.



Durante el copiado es necesario indicar la posición de inicio en el array fuente a partir de la cual los datos empiezan a copiarse al array de destino. En caso del éxito la cantidad de elementos especificada

© 2000-2025, MetaQuotes Ltd.

será copiada desde el array de origen (en este caso se trata del búfer de indicador) al array-receptor. El copiado siempre se realiza tal como se muestra en el dibujo, independientemente de la dirección de indexación establecida en el array-receptor.

Si se supone procesar los datos de precios en un ciclo con el gran número de iteraciones, entonces se recomienda comprobar el hecho de la finalización forzosa del programa utilizando la función IsStopped():

```
int copied=CopyBuffer(ma_handle,// manejador del indicador
                      0,        // índice del búfer de indicador
                      0,        // posición de inicio para el copiado
                      number,   // número de valores a copiar
                      Buffer    // array-receptor de valores
                      );
if(copied<0) return;
int k=0;
while(k<copied && !IsStopped())
  {
   //--- obtenemos el valor para el índice k
   double value=Buffer[k];
   // ...
   // trabajo con el valor value
   k++;
  }
```

Ejemplo:

```
input int per=10; // período del exponente
int ma_handle;    // manejador del indicador
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---
   double ema[10];
   int copied=CopyBuffer(ma_handle,// manejador del indicador
                         0,        // índice del búfer de indicador
                         0,        // posición de inicio para el copiado
                         10,       // número de valores a copiar
```

© 2000-2025, MetaQuotes Ltd.

```
                           ema        // array-receptor de valores
                           );
   if(copied<0) return;
// .... código que sigue
  }
```

Véase también

Organización de acceso a los datos

© 2000-2025, MetaQuotes Ltd.

Organización de acceso a los datos

En esta sección vamos a estudiar las cuestiones relacionadas con la obtención, almacenamiento y solicitud de datos de precios (series temporales).

Recepción de datos del servidor comercial

Antes de que los datos de precios estén disponibles en el terminal MetaTrader 5, es necesario obtener y procesarlos. Para recibir los datos hay que conectarse al terminal comercial MetaTrader 5. Los datos llegan del servidor a petición del terminal en forma de unos bloques de barras de un minuto bien empaquetados.

El mecanismo de dirigirse al servidor para obtener los datos no depende del modo de presentar la solicitud, sea por el usuario navegando por el gráfico o sea de modo de programación en el lenguaje MQL5.

Almacenamiento de datos intermedios

Los datos recibidos del servidor se despaquetan automáticamente y se guardan en el formato intermedio especial HCC. Los datos de cada símbolo se colocan en una carpeta individual *directorio_de_terminal*\bases*nombre_de_servidor*\history*nombre_de_símbolo*. Por ejemplo, los datos del símbolo EURUSD recibidos del servidor comercial MetaQuotes-Demo estarán guardados en la carpeta *directorio_de_terminal*\bases\MetaQuotes-Demo\history\EURUSD\.

Los datos se escriben en los archivos con la extensión .hcc, cada archivo almacena datos de barras de minutos de un año. Por ejemplo, el archivo 2009.hcc de la carpeta EURUSD contiene datos de minutos del símbolo EURUSD del año 2009. Estos archivos se usan para la preparación de datos de precios de todos los períodos y no están destinados para el acceso directo.

Obtención de datos de un período necesario desde los datos intermedios

Los archivos auxiliares en el formato HCC desempeñan el papel de la fuente de información a la hora de construir los datos de precios para los períodos de tiempo solicitados en el formato HC. Los datos en el formato HC son series temporales que están preparadas al máximo para el acceso rápido. Se crean únicamente a la petición de un gráfico o un programa mql5 en el volumen que no supera el valor del parámetro "Max bars in charts" y se guardan en los archivos con la extensión hc para su posterior uso.

Para ahorrar los recursos, los datos relacionados con un período de tiempo se cargan y se almacenan en la memoria operativa sólo si es preciso, en caso si éstos no son requeridos durante un plazo de tiempo considerable se realiza su descarga de la memoria operativa y ellos se guardan en el archivo. Para cada período de tiempo los datos se preparan independientemente de la presencia de datos ya preparados para otros períodos. Las reglas de formación y accesibilidad de datos son iguales para todos los períodos de tiempo. Es decir, a pesar de que la unidad de almacenamiento de datos en el formato HCC sea una barra de un minuto, la disponibilidad de datos en el formato HCC no significa la disponibilidad y accesibilidad de datos del período M1 con el formato HC en el mismo volumen.

La recepción de nuevos datos desde el servidor provoca la renovación automática de datos de precios utilizados en el formato HC de todos los períodos. Esto también lleva al recálculo de todos los indicadores que los usan implícitamente como datos de entrada para el cálculo.

© 2000-2025, MetaQuotes Ltd.

Parámetro "Max bars in chart"

El parámetro "Max bars in charts" limita la cantidad de barras en el formato HC disponible para los gráficos, indicadores y programas mql5. Esta limitación concierne a los datos de todos los períodos de tiempo, y sirve principalmente para ahorrar recursos del ordenador.

Al establecer los valores altos del dicho parámetro, tenemos que recordar que tratándose de un historial bastante profundo de datos de precios para los períodos temporales menores, el consumo de memoria para almacenar series temporales y buffers de indicadores puede ser de centenares de megabytes, llegando al límite de memoria operativa para el terminal de cliente (2GB para las aplicaciones MS Windows de 32 bits).

Los cambios del parámetro "Max bars in charts" tendrán su efecto una vez reiniciado el terminal de cliente. De por sí el cambio de dicho parámetro no implica ni la llamada automática al servidor a por los datos adicionales, ni la formación de barras adicionales de una serie temporal. La solicitud de datos de precios adicionales y la renovación de series temporales se hacen en caso de desplazar el gráfico en la zona de datos que faltan, o bien, solicitando los datos que faltan desde un programa mql5.

El volumen de datos solicitados al servidor corresponde a la cantidad requerida de barras de dicho período, teniendo en cuenta el valor del parámetro "Max bars in charts". La limitación puesta por el parámetro no es tan rigurosa, y en algunas ocasiones el número de barras disponibles en el período temporal puede superar insignificadamente el valor del parámetro corriente.

Disponibilidad de datos

La presencia de datos en el formato HCC, o incluso en el formato HC listo para utilizarse, no siempre supone la disponibilidad absoluta de estos datos para ser mostrados en un gráfico o para utilizarlos en programas mql5.

A la hora de acceder a los datos de precios o a los valores de indicadores desde los programas mql5, hay que recordar que su disponibilidad en un momento dado o desde un momento dado no está garantizada. Esto está relacionado con lo siguiente: con el fin de ahorrar los recursos en MetaTrader 5, no se almacena una copia completa de datos requeridos para un programa mql5, sino se proporciona un acceso directo a la base de datos del terminal.

El historial de precios para todos los períodos temporales se forma de datos comunes en el formato HCC y cualquier renovación de datos desde el servidor supone la renovación de datos para todos los períodos temporales y recálculo de indicadores. A consecuencia de esto, el acceso a los datos puede estar bloqueado, incluso si éstos estaban disponibles hace un momento.

Sincronización de datos del terminal y datos del servidor

Puesto que un programa mql5 puede dirigirse a los datos por cualquier símbolo y período de tiempo, cabe posibilidad que los datos de la serie temporal requerida todavía no están formados en el terminal o los datos de precios requeridos no están sincronizados con el servidor comercial. En este caso es muy complicado pronosticar el tiempo de espera.

Los algoritmos que usan los ciclos de latencia no es la mejor solución. La única excepción en este caso son los scripts, puesto que ellos no tienen otra elección de algoritmo debido a no disponer del procesamiento de eventos. Para los indicadores personalizados dichos algoritmos, así como otros

© 2000-2025, MetaQuotes Ltd.

ciclos de latencia, no se recomiendan en absoluto porque llevan a parar el cálculo de todos los indicadores y otro procesamiento de datos de precios para dicho símbolo.

Para los Asesores Expertos e indicadores personalizados es mejor usar el modelo de eventos de procesamiento. Si durante el manejo del evento OnTick() o OnCalculate() no hemos llegado a obtener todos los datos necesarios de la serie temporal requerida, hay que salir del manejador de eventos, esperando que, al invocar el manejador la próxima vez, obtengamos el acceso a los datos.

Ejemplo de un script para descargar el historial

Vamos a ver un ejemplo - un script ejecuta la solicitud de recibir el historial acerca de un instrumento especificado desde el servidor comercial. Este script sirve para inicializar el instrumento necesario en el gráfico, el período de tiempo no importa, puesto que, y como se decía antes, los datos de precios llegan del servidor en forma de unos datos de minutos empaquetados, de los cuales, luego, se construye una serie temporal predeterminada.

Vamos a organizar todas las acciones de recepción de datos a través de una función independiente CheckLoadHistory(symbol, timeframe, start_date):

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
  }
```

La función CheckLoadHistory() está pensada como una función universal a la que se puede llamar desde un programa cualquiera (Asesor Experto, script o indicador), por tanto hace falta tres parámetros de entrada: nombre del símbolo, período y fecha de inicio a partir de la cual necesitamos el historial de precios.

Vamos a insertar en el código de la función todas las comprobaciones necesarias antes de solicitar el historial que nos falta. Antes de toda hay que asegurarse que el nombre del símbolo y valor del período son correctos:

```
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
```

En el siguiente paso nos persuadimos de que el símbolo especificado esté disponible en la ventana MarketWatch, es decir, el historial para este símbolo va a estar disponible cuando se presenta la solicitud al servidor comercial. Si éste no se encuentra en dicha ventana, hay que añadirlo usando la función SymbolSelect().

```
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
```

Ahora hace falta recibir la fecha de inicio del historial ya disponible para el par símbolo/período especificado. Es posible que el valor del parámetro de entrada startdate, pasado a la función CheckLoadHistory(), entre en el intervalo del historial ya disponible, entonces no hace falta presentar ninguna solicitud al servidor comercial. En este momento la función SeriesInfoInteger() con el modificador SERIES_FIRSTDATE sirve para obtener la primera fecha para el símbolo/período.

```
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
```

© 2000-2025, MetaQuotes Ltd.

Otra verificación importante es la comprobación del tipo de programa desde el cual la función es invocada. Acordemos que el envío de la solicitud para actualizar la serie temporal con el mismo período que tiene el indicador que llama esta actualización es muy indeseable. Esta indeseabilidad está condicionada con el hecho de que la actualización de datos históricos se realice en el mismo hilo en el que opera el indicador. Por eso la posibilidad de clinch es alta. Para la comprobación vamos a usar la función MQL5InfoInteger() con el modificador MQL5_PROGRAM_TYPE.

```
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Syr
      return(-4);
```

Si hemos pasado todas las comprobaciones con éxito, vamos a hacer el último intento de evitar acudir al servidor comercial. Primero averiguaremos la fecha de inicio para la que estén disponibles los datos de minuto en el formato HCC. Vamos a solicitar este valor usando la función SeriesInfoInteger() con el modificador SERIES_TERMINAL_FIRSTDATE, y volvemos a compararlo con el valor del parámetro start_date.

```
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- there is loaded data to build timeseries
      if(first_date>0)
        {
         //--- force timeseries build
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- check date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
```

Si después de todas las comprobaciones el hilo de ejecución sigue estando en el cuerpo de la función CheckLoadHistory(), esto quiere decir que hay necesidad de solicitar los datos de precios que faltan al servidor comercial. Para empezar averiguaremos el valor "Max bars in chart" usando la función TerminalInfoInteger():

```
  int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

Lo vamos a necesitar para no solicitar los datos de más. Luego averiguaremos la primera fecha en el historial del símbolo en el servidor comercial (sin tener en cuenta el período) mediante la función ya conocida SeriesInfoInteger() con el modificador SERIES_SERVER_FIRSTDATE.

```
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
```

Puesto que la solicitud es una operación asincrónica, la función se invoca en el ciclo con un pequeño retraso de 5 milisegundos hasta que la variable first_server_date adquiera un valor, o la ejecución del ciclo sea interrumpida por el usuario (IsStopped(), en este caso devuelve el valor true). Vamos a indicar un valor correcto de la fecha de inicio, desde la cual empezamos a solicitar los datos de precios en el servidor comercial.

```
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Warning: first server date ",first_server_date,
            " for ",symbol," does not match to first series date ",first_date);
```

© 2000-2025, MetaQuotes Ltd.

Si de pronto la fecha de inicio *first_server_date* del servidor resulta ser menos que la fecha de inicio *first_date* del símbolo en el formato HCC, en el diario de registro aparecerá el aviso correspondiente.

Ahora estamos preparados a solicitar los datos de precios que nos faltan al servidor comercial. Vamos a presentar la solicitud en forma del ciclo y empezaremos a rellenar su cuerpo:

```
   while(!IsStopped())
     {
      //1. esperar la sincronización entre la serie temporal reconstruida y el histori
      //2. recibir el número corriente de barras en esta serie temporal
      //   si la cantidad de barras supera el valor de Max_bars_in_chart, podemos sali
      //3. obtenemos la fecha de inicio first_date en la serie temporal reconstruida y
      //   start_date si first_date es menos que start_date, podemos salir, el trabajo
      //4. Solicitamos al servidor comercial nueva parte del historial de 100 barras e
      //   barra disponible numerada "bars"
     }
```

Los tres primeros puntos se implementan por medios ya conocidos.

```
   while(!IsStopped())
     {
      //--- 1.esperamos que se termine el proceso de reconstrucción de la serie tempor
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- 2.preguntamos cuántas barras tenemos disponibles
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         //--- el número de baras es superior a lo que podemos mostrar en el gráfico,
         if(bars>=max_bars) return(-2);
         //--- 3. averiguamos la fecha de inicio corriente en la serie temporal
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            // la fecha de inicio es más temprana que la solicitada, tarea cumplida
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //4. Solicitamos al servidor comercial nueva parte del historial de 100 barras
      //   empezando desde la última barra disponible numerada "bars"
     }
```

Nos queda el último cuarto punto, que es la misma solicitud del historial. No podemos dirigirnos al servidor directamente pero cualquier función-Copy automáticamente inicia el envío de tal solicitud del terminal al servidor comercial si el historial en el formato HCC no es suficiente. Ya que el tiempo de la primera fecha de inicio en la variable *first_date* es el criterio más fácil y natural para evaluar el grado de ejecución de la solicitud, lo más fácil será usar la función CopyTime().

Cuando llamamos a las funciones que se encargan de copiar cualquier dato desde las series temporales, hay que tener en cuenta que el parámetro *start* (número de la barra a partir de la cual se inicia el copiado de datos de precios) siempre tiene que estar dentro de los límites del historial del terminal disponible. Si disponemos sólo de 100 barras, no tiene sentido intentar copiar 300 barras empezando de la barra con el índice 500. Tal solicitud se entenderá como errónea y no será procesada, es decir, no se cargará ningún historial desde el servidor comercial.

Precisamente por eso vamos a copiar 100 barras de una vez, empezando de la barra con el índice bars. Esto proporciona la carga fluida del historial desde el servidor comercial. En realidad se cargará un poco más de 100 barras solicitadas, es que el servidor envía el historial con una cantidad de información ligeramente sobrepasada.

© 2000-2025, MetaQuotes Ltd.

```
   int copied=CopyTime(symbol,period,bars,100,times);
```

Después de la operación de copiado hay que analizar el número de elementos copiados. Si el intento ha sido fallido, el valor de la variable *copied* será igual a cero y el valor del contador fail_cnt será aumentado a 1. El trabajo de la función será detenido después de 100 intentos fallidos.

```
int fail_cnt=0;
...
   int copied=CopyTime(symbol,period,bars,100,times);
   if(copied>0)
     {
      //--- comprobamos los datos
      if(times[0]<=start_date)  return(0);  // el valor copiado es menos, listo
      if(bars+copied>=max_bars) return(-2); // hay más barras que cabe en el gráfico,
      fail_cnt=0;
     }
   else
     {
      //--- no más de 100 intentos fallidos seguidos
      fail_cnt++;
      if(fail_cnt>=100) return(-5);
      Sleep(10);
     }
```

De esta manera, en la función no sólo está implementado el correcto procesamiento de la situación corriente en cada momento de ejecución, sino también se devuelve el código de finalización, el que podemos manejar después de invocar la función CheckLoadHistory() con el fin de obtener información adicional. Por ejemplo, de esta manera:

```
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Símbolo desconocido ",InpLoadedSymbol);
      case -2 : Print("Cantidad superior de barras solicitadas a la que puede ser most
      case -3 : Print("Ejecución interrumpida por el usuario");
      case -4 : Print("Indicador no debe cargar sus propios datos");
      case -5 : Print("Carga fallida");
      case  0 : Print("Todos los datos están cargados");
      case  1 : Print("Cantidad de datos ya disponibles en la serie temporal es sufici
      case  2 : Print("Serie temporal está construida con los datos disponibles en el
      default : Print("Resultado de ejecución no determinado");
     }
```

El código entero de la función viene en el ejemplo del script que demuestra el modo correcto de organizar el acceso a cualquier dato con el procesamiento del resultado de solicitud.

Código:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                TestLoadHistory.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
//--- input parameters
input string          InpLoadedSymbol="NZDUSD";   // Symbol to be load
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1;  // Period to be load
input datetime        InpStartDate=D'2006.01.01'; // Start date
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   Print("Start load",InpLoadedSymbol+","+GetPeriodName(InpLoadedPeriod),"from",InpSta
//---
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Unknown symbol ",InpLoadedSymbol);              break;
      case -2 : Print("Requested bars more than max bars in chart"); break;
      case -3 : Print("Program was stopped");                         break;
      case -4 : Print("Indicator shouldn't load its own data");       break;
      case -5 : Print("Load failed");                                 break;
      case  0 : Print("Loaded OK");                                   break;
      case  1 : Print("Loaded previously");                           break;
      case  2 : Print("Loaded previously and built");                 break;
      default : Print("Unknown result");
     }
//---
   datetime first_date;
   SeriesInfoInteger(InpLoadedSymbol,InpLoadedPeriod,SERIES_FIRSTDATE,first_date);
   int bars=Bars(InpLoadedSymbol,InpLoadedPeriod);
   Print("First date",first_date,"-",bars,"bars");
//---
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
   datetime first_date=0;
   datetime times[100];
//--- check symbol & period
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
//--- check if symbol is selected in the MarketWatch
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
//--- check if data is present
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
//--- don't ask for load of its own data if it is an indicator
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Syr
```

© 2000-2025, MetaQuotes Ltd.

```
      return(-4);
//--- second attempt
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- there is loaded data to build timeseries
      if(first_date>0)
        {
         //--- force timeseries build
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- check date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
//--- max bars in chart from terminal options
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- load symbol history info
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
//--- fix start date for loading
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Warning: first server date ",first_server_date,
```

© 2000-2025, MetaQuotes Ltd.

```
            " for ",symbol," does not match to first series date ",first_date);
//--- load data step by step
   int fail_cnt=0;
   while(!IsStopped())
     {
      //--- wait for timeseries build
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- ask for built bars
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         if(bars>=max_bars) return(-2);
         //--- ask for first date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //--- copying of next part forces data loading
      int copied=CopyTime(symbol,period,bars,100,times);
      if(copied>0)
        {
         //--- check for data
         if(times[0]<=start_date)  return(0);
         if(bars+copied>=max_bars) return(-2);
         fail_cnt=0;
        }
      else
        {
         //--- no more than 100 failed attempts
         fail_cnt++;
         if(fail_cnt>=100) return(-5);
         Sleep(10);
        }
     }
//--- stopped
   return(-3);
  }
//+------------------------------------------------------------------+
//| devuelve a la cadena valor del período                           |
//+------------------------------------------------------------------+
string GetPeriodName(ENUM_TIMEFRAMES period)
  {
   if(period==PERIOD_CURRENT) period=Period();
//---
   switch(period)
     {
      case PERIOD_M1:  return("M1");
      case PERIOD_M2:  return("M2");
      case PERIOD_M3:  return("M3");
      case PERIOD_M4:  return("M4");
      case PERIOD_M5:  return("M5");
      case PERIOD_M6:  return("M6");
      case PERIOD_M10: return("M10");
      case PERIOD_M12: return("M12");
      case PERIOD_M15: return("M15");
      case PERIOD_M20: return("M20");
      case PERIOD_M30: return("M30");
      case PERIOD_H1:  return("H1");
      case PERIOD_H2:  return("H2");
      case PERIOD_H3:  return("H3");
      case PERIOD_H4:  return("H4");
```

© 2000-2025, MetaQuotes Ltd.

```
      case PERIOD_H6:  return("H6");
      case PERIOD_H8:  return("H8");
      case PERIOD_H12: return("H12");
      case PERIOD_D1:  return("Daily");
      case PERIOD_W1:  return("Weekly");
      case PERIOD_MN1: return("Monthly");
     }
//---
   return("unknown period");
  }
```

© 2000-2025, MetaQuotes Ltd.

SeriesInfoInteger

Devuelve la información sobre el estado de datos históricos. Existen 2 variantes de la función.

Directamente devuelve el valor de la propiedad.

```
long  SeriesInfoInteger(
   string                      symbol_name,     // nombre del símbolo
   ENUM_TIMEFRAMES             timeframe,       // período
   ENUM_SERIES_INFO_INTEGER    prop_id,         // identificador de la propiedad
   );
```

Devuelve true o false dependiendo del éxito de ejecución de la función.

```
bool  SeriesInfoInteger(
   string                      symbol_name,     // nombre del símbolo
   ENUM_TIMEFRAMES             timeframe,       // período
   ENUM_SERIES_INFO_INTEGER    prop_id,         // identificador de la propiedad
   long&                       long_var         // variable para recibir la información
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

prop_id

[in] Identificador de la propiedad solicitada, valor de la enumeración ENUM_SERIES_INFO_INTEGER.

long_var

[out] Variable en la que se coloca el valor de la propiedad solicitada.

Valor devuelto

En primer caso se devuelve el valor del tipo long.

Para el segundo caso, la función devuelve true, si dicha propiedad está disponible y su valor ha sido colocado en la variable long_var, de lo contrario devuelve false. Para obtener información adicional sobre un error, hay que llamar a la función GetLastError().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//---
   Print("Cantidad total de barras para el símbolo-período en este momento = ",
         SeriesInfoInteger(Symbol(),Period(),SERIES_BARS_COUNT));

   Print("La primera fecha para el símbolo-período en este momento = ",
         (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_FIRSTDATE));

   Print("La primera fecha en el historial del servidor para el símbolo = ",
         (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_SERVER_FIRSTDATE));

   Print("Datos del símbolo están sincronizados = ",
         (bool)SeriesInfoInteger(Symbol(),Period(),SERIES_SYNCHRONIZED));
  }
```

© 2000-2025, MetaQuotes Ltd.

Bars

Devuelve el número de barras del historial para el símbolo y período correspondientes. Existen 2 variantes de la función.

Solicitar el número de barras en el historial

```
int  Bars(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   );
```

Solicitar el número de barras en el intervalo establecido

```
int  Bars(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   datetime         start_time,       // fecha y hora de inicio
   datetime         stop_time         // fecha y hora de finalización
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

Valor devuelto

Si los parámetros start_time y stop_time están especificados, la función devolverá el número de barras en el rango de fechas. Si estos parámetros no están especificados, la función devolverá el número total de barras.

Nota

Si en el momento cuando se llama a la función Bars(), los datos para la serie temporal con los parámetros especificados todavía no están formados en el terminal, o los datos de la serie temporal no están sincronizados con el servidor comercial a la hora de invocar la función, esta función devolverá el valor cero.

Al solicitar el número de barras en un diapasón de fechas establecido, solo se tienen en cuenta aquellas barras cuya hora de apertura entre en este diapasón. Por ejemplo, si el sábado es el actual día de la semana, entonces, al solicitar el número de barras semanales indicando start_time=último_martes y stop_time=último_viernes, la función retornará 0, dado que la hora de apertura en el marco temporal de una semana siempre cae en domingo, y ni una sola barra de semana entra en el diapasón indicado.

© 2000-2025, MetaQuotes Ltd.

Ejemplo de solicitud del número de barras en la historia:

```
int bars=Bars(_Symbol,_Period);
if(bars>0)
  {
   Print("Cantidad de barras en el historial del terminal para el símbolo-período e
  }
else  //no hay barras disponibles
  {
   //--- por lo visto, datos por el símbolo no están sincronizados con los datos de
   bool synchronized=false;
   //--- contador del ciclo
   int attempts=0;
   // hagamos 5 intentos de esperar la sincronización
   while(attempts<5)
     {
      if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCHRONIZED))
        {
         //--- sincronización con éxito, salimos
         synchronized=true;
         break;
        }
      //--- aumentamos el contador
      attempts++;
      //--- esperaremos 10 milisegundos hasta la siguiente iteración
      Sleep(10);
     }
   //--- salimos del ciclo después de sincronización
   if(synchronized)
     {
      Print("Cantidad de barras en el historial del terminal para el símbolo-períoc
      Print("La primera fecha en el historial del terminal para el símbolo-período
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
      Print("La primera fecha en el historial del servidor para el símbolo = ",
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
     }
   //--- no se ha conseguido la sincronización de datos
   else
     {
      Print("No se ha podido obtener la cantidad de barras para ",_Symbol);
     }
  }
```

Ejemplo de solicitud del número de barras en un intervalo establecido:

```
int n;
datetime date1 = D'2016.09.02 23:55'; // viernes
datetime date2 = D'2016.09.05 00:00'; // lunes
datetime date3 = D'2016.09.08 00:00'; // jueves
//---
n=Bars(_Symbol,PERIOD_H1,D'2016.09.02 02:05',D'2016.09.02 10:55');
Print("Número de barras: ",n); // Mostrará "Número de barras: 8", en el cálculo se
n=Bars(_Symbol,PERIOD_D1,date1,date2);
Print("Número de barras: ",n); // Mostrará "Número de barras: 1", puesto que en el
n=Bars(_Symbol,PERIOD_W1,date2,date3);
Print("Número de barras: ",n); // Mostrará "Número de barras: 0", puesto que en el
```

Véase también

© 2000-2025, MetaQuotes Ltd.

Funciones de procesamiento de eventos

© 2000-2025, MetaQuotes Ltd.

BarsCalculated

Devuelve la cantidad de datos calculados para el indicador especificado.

```
int  BarsCalculated(
   int       indicator_handle,     // manejador del indicador
   );
```

Parámetros

indicator_handle

[in] Manejador del indicador obtenido por la función de indicador correspondiente.

Valor devuelto

Devuelve la cantidad de datos calculados en el búfer de indicadores, o devuelve -1 en caso del error (datos aún no han sido calculados).

Nota

Esta función es útil cuando es necesario conseguir los datos de indicador inmediatamente después de su creación (recepción del manejador del indicador).

Ejemplo:

```
void OnStart()
  {
   double Ups[];
//--- establecemos para los arrays el indicio de una serie temporal
   ArraySetAsSeries(Ups,true);
//--- creamos el manejador del indicador Fractals
   int FractalsHandle=iFractals(NULL,0);
//--- reinicializamos el código del error
   ResetLastError();
//--- intentaremos copiar el valor del indicador
   int i,copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Sleep(50);
      for(i=0;i<100;i++)
        {
         if(BarsCalculated(FractalsHandle)>0)
            break;
         Sleep(50);
        }
      copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
      if(copied<=0)
        {
         Print("Fallo al copiar fractales superiores. Error = ",GetLastError(),
         "i = ",i,"    copied = ",copied);
         return;
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      else
        Print("Los fractales superiores se han copiado con éxito.",
        "i = ",i,"    copied = ",copied);
    }
  else Print("Los fractales superiores se han copiado con éxito. ArraySize = ",ArrayS
}
```

© 2000-2025, MetaQuotes Ltd.

IndicatorCreate

Devuelve el manejador del indicador técnico especificado que ha sido creado a base del array de parámetros del tipo MqlParam.

```
int  IndicatorCreate(
   string           symbol,                    // nombre del símbolo
   ENUM_TIMEFRAMES  period,                    // período
   ENUM_INDICATOR   indicator_type,            // tipo del indicador desde la
   int              parameters_cnt=0,          // número de parámetros
   const MqlParam&  parameters_array[]=NULL,   // array de parámetros
   );
```

Parámetros

symbol

[in] Nombre de símbolo del instrumento, a base de los datos de la cual se calcula el indicador. NULL significa el símbolo actual.

period

[in] Valor del período de tiempo puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el período de tiempo corriente.

indicator_type

[in] Tipo del indicador, puede ser uno de los valores de la enumeración ENUM_INDICATOR.

parameters_cnt

[in] Número de parámetros pasados en el array parameters_array[]. Los elementos del array tienen un tipo especial de la estructura MqlParam. Por defecto, valor cero - los parámetros no se pasan. Si el número de parámetros indicado es distinto a cero, entonces el parámetro *parameters_array* es obligatorio. Se puede pasar no más de 64 parámetros.

parameters_array[]=NULL

[in] Array del tipo MqlParam, cuyos elementos contienen el tipo y valor de cada uno de los parámetros de entrada del indicador técnico.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE.

Nota

Si se crea el manejador del indicador del tipo IND_CUSTOM, el campo *type* del primer elemento del array de los parámetros de entrada *parameters_array* obligatoriamente tiene que tener el valor TYPE_STRING desde la enumeración ENUM_DATATYPE, y el campo *string_value* del primer elemento tiene que tener el nombre del indicador personalizado. El indicador personalizado tiene que estar compilado (archivo con la extensión EX5) y ubicarse en el directorio MQL5/Indicators del terminal de cliente o en una subcarpeta.

Los indicadores que requieren verificaciones se definen automáticamente desde la llamada a la función iCustom(), si el parámetro correspondiente es fijado por una cadena constante. Para los

© 2000-2025, MetaQuotes Ltd.

demás casos (uso de función IndicatorCreate() o uso de una cadena no constante en el parámetro que asigna el nombre del indicador) esta propiedad #property tester_indicator es necesaria:

```
#property tester_indicator "indicator_name.ex5"
```

Si en un indicador personalizado se utiliza la primera forma de la llamada, entonces a la hora de pasar los parámetros de entrada, a través del último parámetro se puede especificar adicionalmente a base de qué datos éste va a ser calculado. Si el parámetro "Apply to" no está especificado explícitamente, por defecto el cálculo se basa en los valores PRICE_CLOSE.

Ejemplo:

```
void OnStart()
  {
   MqlParam params[];
   int      h_MA,h_MACD;
//--- create iMA("EURUSD",PERIOD_M15,8,0,MODE_EMA,PRICE_CLOSE);
   ArrayResize(params,4);
//--- set ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=8;
//--- set ma_shift
   params[1].type         =TYPE_INT;
   params[1].integer_value=0;
//--- set ma_method
   params[2].type         =TYPE_INT;
   params[2].integer_value=MODE_EMA;
//--- set applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=PRICE_CLOSE;
//--- create MA
   h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);
//--- create iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA);
   ArrayResize(params,4);
//--- set fast ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=12;
//--- set slow ma_period
   params[1].type         =TYPE_INT;
   params[1].integer_value=26;
//--- set smooth period for difference
   params[2].type         =TYPE_INT;
   params[2].integer_value=9;
//--- set indicator handle as applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=h_MA;
//--- create MACD based on moving average
   h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- use indicators
//--- . . .
//--- release indicators (first h_MACD)
   IndicatorRelease(h_MACD);
   IndicatorRelease(h_MA);
  }
```

© 2000-2025, MetaQuotes Ltd.

IndicatorParameters

Devuelve para el manejador especificado el número de los parámetros de entrada del indicador, así como los propios valores y el tipo de parámetros.

```
int  IndicatorParameters(
   int               indicator_handle,     // manejador del indicador
   ENUM_INDICATOR&   indicator_type,       // variable para recibir el tipo del indica
   MqlParam&         parameters[]          // array para recibir parámetros
   );
```

Parámetros

indicator_handle

[in] Manejador del indicador para el que hace falta averiguar el número de parámetros sobre los cuales está calculado éste.

indicator_type

[out] Variable del tipo ENUM_INDICATOR en la que será escrito el tipo del indicador.

parameters[]

[out] Array dinámico para recibir los valores del tipo MqlParam en el que será escrita la lista de parámetros del indicador. El tamaño del array es devuelto por la misma función IndicatorParameters().

Valor devuelto

El número de los parámetros de entrada del indicador con el manejador especificado. En caso del error devuelve -1. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {

//--- número de ventanas sobre el gráfico (siempre hay por lo menos una ventana princi
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
//--- vamos por las ventanas del gráfico
   for(int w=0;w<windows;w++)
     {
      //--- número de indicadores en esta ventana/subventana
      int total=ChartIndicatorsTotal(0,w);
      //--- repasaremos todos los indicadores en la ventana
      for(int i=0;i<total;i++)
        {
         //--- obtenemos el nombre corto del indicador
         string name=ChartIndicatorName(0,w,i);
         //--- obtenemos el manejador del indicador
```

© 2000-2025, MetaQuotes Ltd.

```
         int handle=ChartIndicatorGet(0,w,name);
         //--- mostramos en el diario
         PrintFormat("Window=%d,  indicator #%d,  handle=%d",w,i,handle);
         //---
         MqlParam parameters[];
         ENUM_INDICATOR indicator_type;
         int params=IndicatorParameters(handle,indicator_type,parameters);
         //--- encabezamiento del mensaje
         string par_info="Short name "+name+", type "
                         +EnumToString(ENUM_INDICATOR(indicator_type))+"\r\n";
         //---
         for(int p=0;p<params;p++)
           {
            par_info+=StringFormat("parameter %d: type=%s, long_value=%d, double_value
                                   p,
                                   EnumToString((ENUM_DATATYPE)parameters[p].type),
                                   parameters[p].integer_value,
                                   parameters[p].double_value,
                                   parameters[p].string_value
                                   );
           }
         Print(par_info);
        }
      //--- hemos pasado por todos los indicadores en la ventana
     }
//---
  }
```

Véase también

ChartIndicatorGet()

© 2000-2025, MetaQuotes Ltd.

IndicatorRelease

Elimina el manejador del indicador y libera la parte calculadora del indicador si nadie la usa.

```
bool  IndicatorRelease(
   int       indicator_handle     // manejador del indicador
   );
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

Esta función permite eliminar el manejador del indicador si no se necesita más, y así permite ahorrar el espacio de la memoria. El manejador se elimina inmediatamente, mientras que la parte calculadora del indicador se elimina dentro de un rato (si no hay más llamadas a ésta).

Al trabajar en el simulador de estrategias, la función IndicatorRelease() no se ejecuta.

Ejemplo:

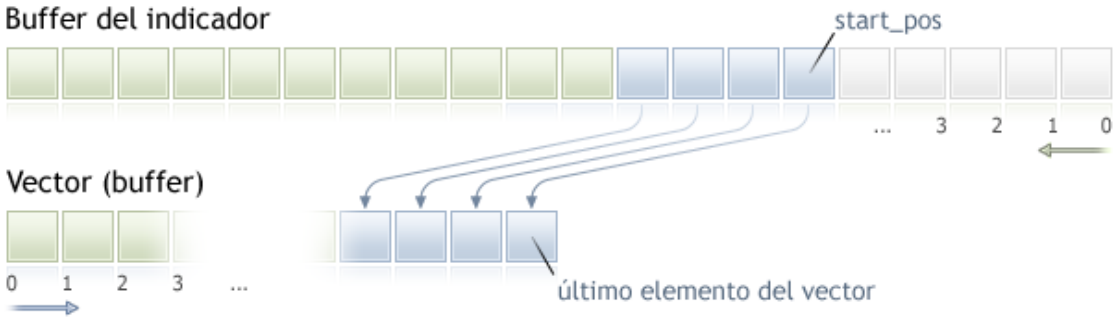
© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                        Test_IndicatorRelease.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- input parameters
input int                 MA_Period=15;
input int                 MA_shift=0;
input ENUM_MA_METHOD      MA_smooth=MODE_SMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
//--- vamos a almacenar el manejador del indicador
int MA_handle;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- creamos el manejador del indicador
   MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,price);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- si el valor de la variable global todavía no está indicado
   if(GlobalVariableCheck("MA_value")==0)
     {
      //--- array dinámico para los valores del indicador
      double v[];
      //--- obtenemos el valor del indicador en dos últimas barras
      if(CopyBuffer(MA_handle,0,0,2,v)==2 && v[1]!=EMPTY_VALUE)
        {
         //--- recordamos en la variable global valor en la penúltima barra
         if(GlobalVariableSet("MA_value",v[1]))
           {
            //--- liberamos el manejador del indicador
            if(!IndicatorRelease(MA_handle))
               Print("IndicatorRelease() failed. Error ",GetLastError());
           }
        }
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

CopyBuffer

Recibe en el array búfer los datos del búfer especificado del indicador indicado en cantidad especificada.



La cuenta de elementos a copiar (búfer de indicadores con el índice buffer_num) desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente (valor del indicador para la barra corriente).

Cuando copiamos los datos sin conocer todavía el volumen a copiar, se recomienda usar un array dinámico como array de destino buffer[], puesto que la función CopyBuffer() trata de ajustar el tamaño del array-receptor en función del espacio que ocupan los datos que se copian. Si un búfer de indicador (array preasignado por la función SetIndexBufer() para almacenar los valores del indicador) se usa como el array-receptor buffer[], entonces se admite el copiado parcial. Podemos ver su ejemplo en el indicador personalizado Awesome_Oscillator.mq5 que entra en el pack estándar del terminal.

Si tenemos que copiar parcialmente los valores del indicador a otro array (que no sea un búfer de indicador), para eso hay que usar un array intermedio donde se copia la cantidad necesaria, y luego de este array-intermediario copiar la cantidad necesaria elemento por elemento a lugares correspondientes del array de destino.

Si sabemos la cantidad de datos que tenemos que copiar, con el fin de evitar la asignación excesiva de memoria es mejor hacerlo a un buffer asignado estáticamente.

La propiedad del array de destino no importa, sea as_series=true o sea as_series=false. Los datos van a ser copiados de tal manera que el elemento más antiguo estará al principio de la memoria física que ha sido destinada para el array. Existen 3 variantes de la función.

Llamada según la posición de inicio y el número de elementos requeridos

```
int  CopyBuffer(
   int       indicator_handle,     // manejador del indicador
   int       buffer_num,           // número del buffer del indicador
   int       start_pos,            // posición de inicio
   int       count,                // cantidad de datos a copiar
   double    buffer[]              // array de destino en el que se copian los datos
   );
```

Llamada según la fecha de inicio y el número de elementos requeridos

```
int  CopyBuffer(
```

© 2000-2025, MetaQuotes Ltd.

```
   int       indicator_handle,     // manejador del indicador
   int       buffer_num,           // número del buffer del indicador
   datetime  start_time,           // fecha y hora de inicio
   int       count,                // cantidad de datos a copiar
   double    buffer[]              // array de destino en el que se copian los datos
   );
```

Llamada según la fecha de inicio y finalización del intervalo de tiempo requerido

```
int  CopyBuffer(
   int       indicator_handle,     // manejador del indicador
   int       buffer_num,           // número del buffer del indicador
   datetime  start_time,           // fecha y hora de inicio
   datetime  stop_time,            // fecha y hora de finalización
   double    buffer[]              // array de destino en el que se copian los datos
   );
```

Parámetros

indicator_handle

[in] Manejador del indicador recibido por la función de indicador correspondiente.

buffer_num

[in] Número del búfer de indicadores.

start_pos

[in] Posición del primer elemento a copiar.

count

[in] Cantidad de elementos a copiar.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

buffer[]

[out] Array del tipo double.

Valor devuelto

Cantidad de elementos copiados del array, o -1 en caso del error.

Nota

Al solicitar los datos del indicador, si las series temporales solicitadas todavía no están construidas o hay que bajarlas del servidor, la función devolverá inmediatamente -1. Aunque en este caso, se iniciará el proceso de descarga/construcción.

Cuando se solicitan los datos a un Asesor Experto o un script, se iniciará la descarga desde el servidor si el terminal no dispone de estos datos a nivel local, o se empezará la construcción de la serie temporal necesaria si se puede construir los datos usando el historial local y ellos todavía no

© 2000-2025, MetaQuotes Ltd.

están listos. La función devolverá aquella cantidad de datos que estarán listos para el momento de vencimiento de timeout.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                              TestCopyBuffer3.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input bool               AsSeries=true;
input int                period=15;
input ENUM_MA_METHOD     smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
input int                shift=0;
//--- indicator buffers
double                   MABuffer[];
int                      ma_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Parámetro AsSeries = ",AsSeries);
   Print("Búfer de indicadores después de SetIndexBuffer() es una serie temporal = ",
         ArrayGetAsSeries(MABuffer));
//--- set short indicator name
   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- set AsSeries(dependes from input parameter)
   ArraySetAsSeries(MABuffer,AsSeries);
   Print("Búfer de indicadores después de ArraySetAsSeries(MABuffer,true); es una seri
         ArrayGetAsSeries(MABuffer));
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- check if all data calculated
   if(BarsCalculated(ma_handle)<rates_total) return(0);
//--- we can copy not all data
   int to_copy;
   if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
   else
     {
      to_copy=rates_total-prev_calculated;
      //--- last value is always copied
      to_copy++;
     }
//--- try to copy
   if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

En el ejemplo de arriba se nos muestra que el búfer de indicadores se rellena con valores de otro búfer de indicadores perteneciente a indicador del mismo símbolo/período.

Puede encontrar un ejemplo más detallado de la solicitud de datos históricos en la sección Modos de enlace de objetos. En el script de aquella sección se muestra cómo se puede obtener los valores del indicador iFractals sobre las últimas 1000 barras, y cómo luego se puede visualizar en el gráfico los diez últimos fractales arriba y los diez último fractales abajo. Esta técnica puede ser utilizada para todos los indicadores que tienen omisiones de valores y suelen realizarse utilizando los siguientes estilos de construcción:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,

© 2000-2025, MetaQuotes Ltd.

- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

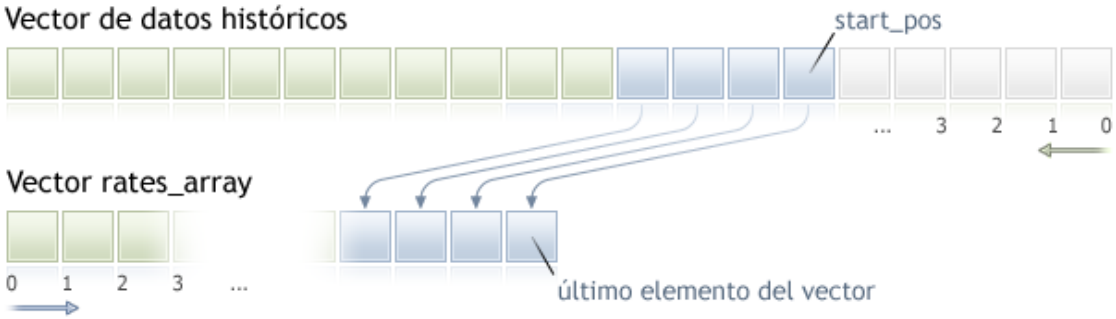
Véase también

Propiedades de indicadores personalizados, SetIndexBuffer

© 2000-2025, MetaQuotes Ltd.

CopyRates

Recibe en el array rates_array datos históricos de la estructura MqlRates del símbolo-período especificado en cantidad especificada. La cuenta de elementos desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente.



Cuando copiamos los datos sin conocer todavía el volumen a copiar, se recomienda usar un array dinámico como array de destino, porque si la cantidad de datos resulta ser menos (o más) de la que cabe en el array, entonces se intenta redistribuirlo de tal manera para que los datos solicitados quepan íntegramente.

Si sabemos la cantidad de datos que tenemos que copiar, con el fin de evitar la asignación excesiva de memoria es mejor hacerlo a un buffer asignado estáticamente.

La cuenta de elementos desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente. Existen 3 variantes de la función.

Llamada según la posición de inicio y el número de elementos requeridos

```
int  CopyRates(
   string           symbol_name,       // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,         // período
   int              start_pos,         // posición de inicio
   int              count,             // cantidad de datos a copiar
   MqlRates         rates_array[]      // array de destino en el que se copian los dat
   );
```

Llamada según la fecha de inicio y el número de elementos requeridos

```
int  CopyRates(
   string           symbol_name,       // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // fecha y hora de inicio
   int              count,             // cantidad de datos a copiar
   MqlRates         rates_array[]      // array de destino en el que se copian los dat
   );
```

Llamada según la fecha de inicio y finalización del intervalo de tiempo requerido

```
int  CopyRates(
```

© 2000-2025, MetaQuotes Ltd.

```
   string           symbol_name,       // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // fecha y hora de inicio
   datetime         stop_time,         // fecha y hora de finalización
   MqlRates         rates_array[]      // array de destino en el que se copian los dat
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_time

[in] Hora de la barra correspondiente al primer elemento.

start_pos

[in] Posición del primer elemento a copiar.

count

[in] Cantidad de elementos a copiar.

stop_time

[in] Hora de la barra correspondiente al último elemento.

rates_array[]

[out] Array del tipo MqlRates.

Valor devuelto

Cantidad de elementos copiados del array, o -1 en caso del error.

Nota

Si el intervalo de datos solicitados se encuentra totalmente fuera del rango de datos disponibles del servidor, la función devuelve -1. Si los datos solicitados salen del rango de TERMINAL_MAXBARS (la cantidad máxima de barras en el gráfico), la función tambien devuelve -1.

Al solicitar los datos del indicador, si las series temporales solicitadas todavía no están construidas o hay que bajarlas del servidor, la función devolverá inmediatamente -1. Aunque en este caso, se iniciará el proceso de descarga/construcción.

Cuando se solicitan los datos a un Asesor Experto o un script, se iniciará la descarga desde el servidor si el terminal no dispone de estos datos a nivel local, o se empezará la construcción de la serie temporal necesaria si se puede construir los datos usando el historial local y ellos todavía no están listos. La función devolverá aquella cantidad de datos que estarán listos para el momento de vencimiento de timeout, pero el historial seguirá cargándose y con la siguiente solicitud del mismo tipo la función devolverá más datos.

Cuando se solicitan los datos por la fecha inicial y el número de elementos requeridos, sólo se devuelven los datos cuya fecha es menor (anterior) o igual a la especificada. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la fecha de apertura de cualquier barra

© 2000-2025, MetaQuotes Ltd.

para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre es igual o menor que la especificada.

Cuando se solicitan los datos de un período de fechas especificado, se devuelven los datos que entran sólo en este intervalo temporal especificado. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la hora de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre se encuentra en el intervalo especificado.

Por ejemplo, si el día corriente es sabado, al intentar copiar los datos del margen semanal indicando *start_time=Último_Martes* y *stop_time=Último_Viernes*, la función devolverá 0 porque la apertura en un período de tiempo semanal siempre cae en domingo, pero ninguna barra semanal no entra en el período especificado.

Si se necesita obtener el valor que corresponde a una barra corriente no finalizada, se puede usar la primera forma de llamada, indicando *start_pos*=0 y *count*=1.

Ejemplo:

```
void OnStart()
  {
//---
   MqlRates rates[];
   ArraySetAsSeries(rates,true);
   int copied=CopyRates(Symbol(),0,0,100,rates);
   if(copied>0)
     {
      Print("Barras copiadas: "+copied);
      string format="open = %G, high = %G, low = %G, close = %G, volume = %d";
      string out;
      int size=fmin(copied,10);
      for(int i=0;i<size;i++)
        {
         out=i+":"+TimeToString(rates[i].time);
         out=out+" "+StringFormat(format,
                                  rates[i].open,
                                  rates[i].high,
                                  rates[i].low,
                                  rates[i].close,
                                  rates[i].tick_volume);
         Print(out);
        }
     }
   else Print("Fallo al recibir datos históricos para el símbolo ",Symbol());
  }
```

Puede encontrar un ejemplo más detallado de la solicitud de datos históricos en la sección Modos de enlace de objetos. En el script de aquella sección se muestra cómo se puede obtener los valores del indicador iFractals sobre las últimas 1000 barras, y cómo luego se puede visualizar en el gráfico los diez últimos fractales arriba y los diez último fractales abajo. Esta técnica puede ser utilizada para

© 2000-2025, MetaQuotes Ltd.

todos los indicadores que tienen omisiones de valores y suelen realizarse utilizando los siguientes estilos de construcción:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

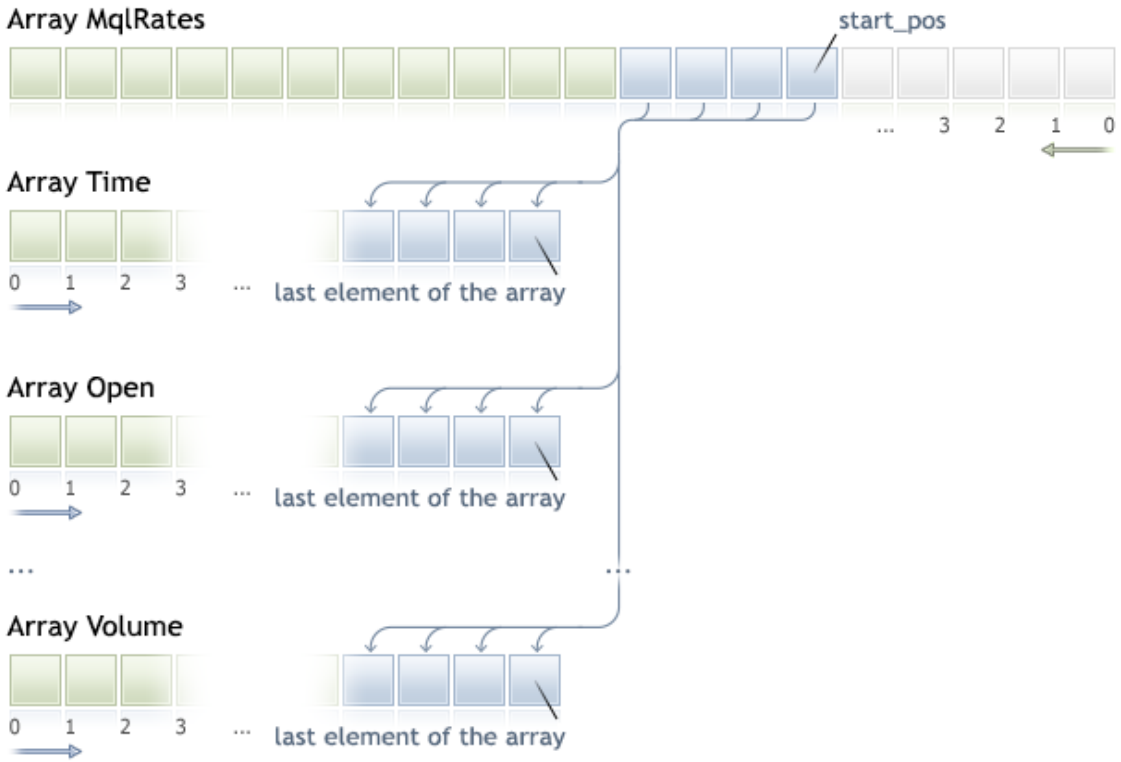
Véase también

Estructuras y clases, TimeToString, StringFormat

© 2000-2025, MetaQuotes Ltd.

CopySeries

Obtiene en el conjunto especificado de arrays las series temporales sincronizadas de la estructura MqlRates para símbolo-periodo indicado en el número especificado. Los elementos se contarán partiendo de la posición inicial desde el presente hasta el pasado, es decir, si la posición inicial es igual a 0, esto indicará la barra actual.



Al copiar una cantidad de datos previamente desconocida, se recomienda usar como array receptor un array dinámico, ya que si hay más datos de los que puede contener el array, se intentará redistribuir el array de tal forma que los datos solicitados se ajusten por completo.

Si necesitamos copiar una cantidad conocida de datos, será mejor hacerlo en un búfer asignado estáticamente para evitar reasignaciones de memoria innecesarias.

No importa qué propiedad tenga el array receptor: as_series=true o as_series=false, los datos se copiarán de tal forma que el elemento de serie temporal más antiguo se copie al inicio de la memoria física asignada para el array.

```
int  CopySeries(
   string           symbol_name,       // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,         // periodo
   int              start_pos,         // desde dónde comenzaremos
   int              count,             // cuánto copiaremos
   ulong            rates_mask,        // combinación de banderas para indicar las se
   void&            array1[],          // array al que se copiarán los datos de la pr
   void&            array2[]           // array al que se copiarán los datos de la se
   ...
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Periodo.

start_pos

[in] Número del primer elemento copiado.

count

[in] Número de elementos copiados.

rates_mask

[in] Combinación de banderas de la enumeración ENUM_COPY_RATES .

array1, array2,...

[out] Array del tipo correspondiente para obtener la serie temporal de la estructura MqlRates. Orden de los arrays transmitidos a la función, debe corresponderse con el orden de los campos en la estructura MqlRates.

Valor retornado

Número de elementos copiados del array, o bien -1 en caso de error.

Observación

Si el intervalo de las series temporales se encuentra totalmente fuera de los datos en el servidor, la función retornará -1. Si los datos solicitados se encuentran fuera de TERMINAL_MAXBARS (número máximo de barras en el gráfico), la función también retornará -1.

Al solicitar los datos desde el indicador, si las series temporales solicitadas aún no han sido construidas o deben ser cargadas desde el servidor, la función retornará de inmediato -1, pero, en este caso, el propio proceso de carga/construcción será iniciado.

Al solicitar los datos desde un experto o script, se inicializará la carga desde el servidor, si el terminal no dispone de estos datos a nivel local; o bien comenzará la construcción de la serie temporal necesaria, si los datos se pueden construir a partir de la historia local, pero aún no están preparados. La función retornará la cantidad de datos que estén preparados al momento de finalización del timeout, pero la carga de la historia continuará, y con la siguiente solicitud análoga, la función retornará ya más datos.

Diferencia entre CopySeries y CopyRates

La función CopySeries permite obtener de una sola vez solo las series temporales necesarias en diferentes arrays especificados; en este caso, además, todas estarán sincronizadas entre sí. Esto significa que todos los valores en los arrays resultantes en un índice N específico pertenecerán a la misma barra en el par Símbolo/Marco temporal especificado. En este caso, el programador no necesitará asegurarse de que todas las series temporales obtenidas estén sincronizadas según la hora de apertura de la barra.

© 2000-2025, MetaQuotes Ltd.

A diferencia de CopyRates, que retorna el conjunto completo de series temporales como una matriz MqlRates, la función CopySeries permite al programador obtener solo las series temporales requeridas como arrays aparte; para ello, deberemos especificar una combinación de banderas para indicar el tipo de series temporales solicitadas. En este caso, el orden de los arrays transmitidos a la función deberá corresponderse con el orden de los campos en la estructura MqlRates:

```
struct MqlRates
  {
   datetime time;         // hora de inicio del periodo
   double   open;         // precio de apertura
   double   high;         // precio mayor del periodo
   double   low;          // precio menor del periodo
   double   close;        // precio de cierre
   long     tick_volume;  // volumen de ticks
   int      spread;       // spread
   long     real_volume;  // volumen bursátil
  }
```

De esta forma, si resulta necesario obtener los valores de las series temporales *time*, *close* y *real_volume* para las últimas 100 barras del Símbolo/Marco temporal actual, la llamada deberá ser la siguiente:

```
datetime time[];
double   close[];
long     volume[];
CopySeries(NULL,0,0,100,COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_VOLUME_REAL,time,c
```

En este caso, será importante el orde de los arrays "*time*, *close*, *volume*", este deberá corresponderse con el orden de los campos en la estructura MqlRates. Por otro lado, el orden de los valores en la máscara *rates_mask* carece de importancia, la máscara podría ser la siguiente:

```
COPY_RATES_VOLUME_REAL|COPY_RATES_TIME|COPY_RATES_CLOSE
```

Ejemplo:

```
//--- input parameters
input datetime InpDateFrom=D'2022.01.01 00:00:00';
input datetime InpDateTo  =D'2023.01.01 00:00:00';
input uint     InpCount   =20;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- arrays para obtener las series temporales de la estructura MqlRates
   double   open[];
   double   close[];
   float    closef[];
   datetime time1[], time2[];
//---solicitamos los precios de cierre en un array de tipo double
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   int res1=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE, time1, close);
   PrintFormat("1. CopySeries  returns %d values. Error code=%d", res1, GetLastError()
   ArrayPrint(close);

//--- ahora solicitamos más precios de apertura, y los precios de cierre los solicitar
   ResetLastError();
   int res2=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_OPEN, time2, open,
   PrintFormat("2. CopySeries  returns %d values. Error code=%d", res2, GetLastError()
   ArrayPrint(closef);
//--- comparamos los datos obtenidos
   if((res1==res2) && (time1[0]==time2[0]))
     {
      Print("  | Time              |    Open      | Close double | Close float |");
      for(int i=0; i<10; i++)
        {
         PrintFormat("%d | %s |   %.5f    |   %.5f    |   %.5f   |",
                     i, TimeToString(time1[i]), open[i], close[i], closef[i]);
        }
     }
//--- Resultado
      1. CopySeries  returns 20 values. Error code=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684 1.(
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239 1.(
      2. CopySeries  returns 20 values. Error code=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684 1.(
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239 1.(
        | Time              |    Open      | Close double | Close float |
      0 | 2023.03.01 17:00 |   1.06660    |   1.06722    |   1.06722   |
      1 | 2023.03.01 18:00 |   1.06722    |   1.06733    |   1.06733   |
      2 | 2023.03.01 19:00 |   1.06734    |   1.06653    |   1.06653   |
      3 | 2023.03.01 20:00 |   1.06654    |   1.06520    |   1.06520   |
      4 | 2023.03.01 21:00 |   1.06520    |   1.06573    |   1.06573   |
      5 | 2023.03.01 22:00 |   1.06572    |   1.06649    |   1.06649   |
      6 | 2023.03.01 23:00 |   1.06649    |   1.06694    |   1.06694   |
      7 | 2023.03.02 00:00 |   1.06683    |   1.06675    |   1.06675   |
      8 | 2023.03.02 01:00 |   1.06675    |   1.06684    |   1.06684   |
      9 | 2023.03.02 02:00 |   1.06687    |   1.06604    |   1.06604   |
//---
  }
```

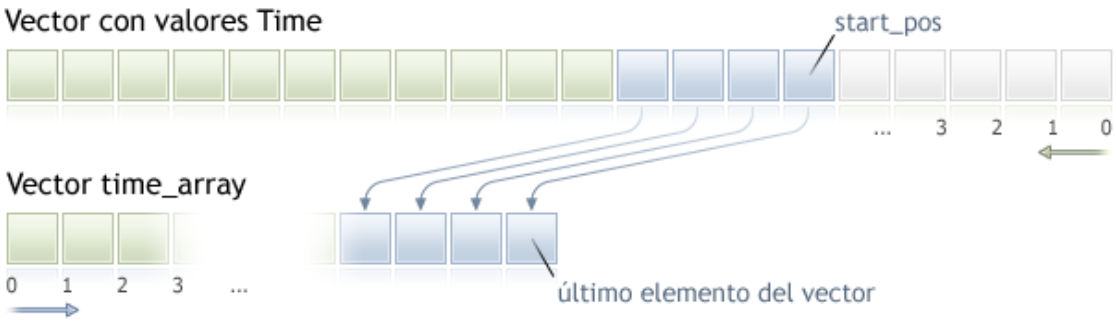
Ver también

Estructuras y clases, CopyRates

© 2000-2025, MetaQuotes Ltd.

CopyTime

La función recibe en el array time_array datos históricos de la hora de apertura de barras para el par especificado de símbolo-período en cantidad especificada. Cabe mencionar que la cuenta de elementos desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente.



Cuando copiamos los datos sin conocer todavía el volumen a copiar, se recomienda usar un array dinámico como array de destino, porque si la cantidad de datos resulta ser menos (o más) de la que cabe en el array, entonces se intenta redistribuirlo de tal manera para que los datos solicitados quepan íntegramente.

Si sabemos la cantidad de datos que tenemos que copiar, con el fin de evitar la asignación excesiva de memoria es mejor hacerlo a un buffer asignado estáticamente.

La propiedad del array de destino no importa, sea as_series=true o sea as_series=false. Los datos van a ser copiados de tal manera que el elemento más antiguo estará al principio de la memoria física que ha sido destinada para el array. Existen 3 variantes de la función.

Llamada según la posición de inicio y el número de elementos requeridos

```
int  CopyTime(
   string           symbol_name,     // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,       // período
   int              start_pos,       // posición de inicio
   int              count,           // cantidad de datos a copiar
   datetime         time_array[]     // array para copiar la hora de apertura
   );
```

Llamada según la fecha de inicio y el número de elementos requeridos

```
int  CopyTime(
   string           symbol_name,     // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,       // período
   datetime         start_time,      // fecha y hora de inicio
   int              count,           // cantidad de datos a copiar
   datetime         time_array[]     // array para copiar la hora de apertura
   );
```

Llamada según la fecha de inicio y finalización del intervalo de tiempo requerido

© 2000-2025, MetaQuotes Ltd.

```
int  CopyTime(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   datetime         start_time,       // fecha y hora de inicio
   datetime         stop_time,        // fecha y hora de finalización
   datetime         time_array[]      // array para copiar la hora de apertura
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_pos

[in] Posición del primer elemento a copiar.

count

[in] Cantidad de elementos a copiar.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

time_array[]

[out] Array del tipo datetime.

Valor devuelto

Cantidad de elementos copiados del array, o -1 en caso del error.

Nota

Si el intervalo de datos solicitados se encuentra totalmente fuera del rango de datos disponibles del servidor, la función devuelve -1. Si los datos solicitados salen del rango de TERMINAL_MAXBARS (la cantidad máxima de barras en el gráfico), la función tambien devuelve -1.

Al solicitar los datos del indicador, si las series temporales solicitadas todavía no están construidas o hay que bajarlas del servidor, la función devolverá inmediatamente -1. Aunque en este caso, se iniciará el proceso de descarga/construcción.

Cuando se solicitan los datos a un Asesor Experto o un script, se iniciará la descarga desde el servidor si el terminal no dispone de estos datos a nivel local, o se empezará la construcción de la serie temporal necesaria si se puede construir los datos usando el historial local y ellos todavía no están listos. La función devolverá aquella cantidad de datos que estarán listos para el momento de vencimiento de timeout, pero el historial seguirá cargándose y con la siguiente solicitud del mismo tipo la función devolverá más datos.

© 2000-2025, MetaQuotes Ltd.

Cuando se solicitan los datos por la fecha inicial y el número de elementos requeridos, sólo se devuelven los datos cuya fecha es menor (anterior) o igual a la especificada. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la fecha de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre es igual o menor que la especificada.

Cuando se solicitan los datos de un período de fechas especificado, se devuelven los datos que entran sólo en este intervalo temporal especificado. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la hora de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre se encuentra en el intervalo especificado.

Por ejemplo, si el día corriente es sabado, al intentar copiar los datos del margen semanal indicando *start_time=Último_Martes* y *stop_time=Último_Viernes*, la función devolverá 0 porque la apertura en un período de tiempo semanal siempre cae en domingo, pero ninguna barra semanal no entra en el período especificado.

Si se necesita obtener el valor que corresponde a una barra corriente no finalizada, se puede usar la primera forma de llamada, indicando *start_pos*=0 y *count*=1.

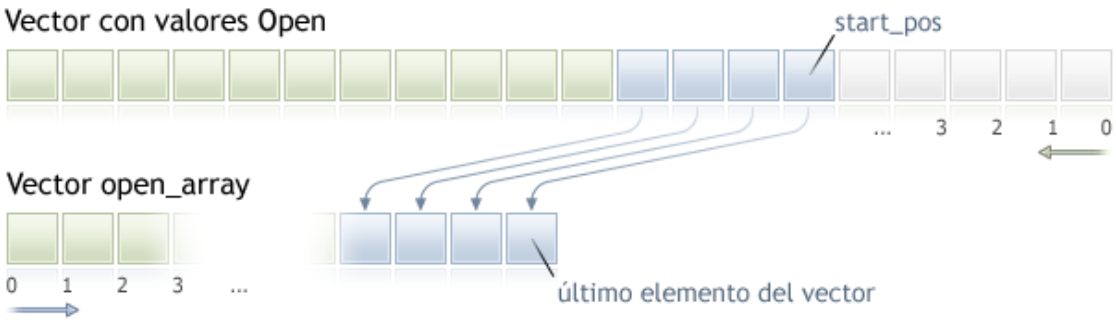
Puede encontrar un ejemplo de la solicitud de datos históricos en la sección Modos de enlace de objetos. En el script de aquella sección se muestra cómo se puede obtener los valores del indicador iFractals sobre las últimas 1000 barras, y cómo luego se puede visualizar en el gráfico los diez últimos fractales arriba y los diez último fractales abajo. Esta técnica puede ser utilizada para todos los indicadores que tienen omisiones de valores y suelen realizarse utilizando los siguientes estilos de construcción:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyOpen

La función recibe en el array open_array datos históricos de los precios de apertura de barras para el par especificado de símbolo-período en cantidad especificada. Cabe mencionar que la cuenta de elementos desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente.



Cuando copiamos los datos sin conocer todavía el volumen a copiar, se recomienda usar un array dinámico como array de destino, porque si la cantidad de datos resulta ser menos (o más) de la que cabe en el array, entonces se intenta redistribuirlo de tal manera para que los datos solicitados quepan íntegramente.

Si sabemos la cantidad de datos que tenemos que copiar, con el fin de evitar la asignación excesiva de memoria es mejor hacerlo a un buffer asignado estáticamente.

La propiedad del array de destino no importa, sea as_series=true o sea as_series=false. Los datos van a ser copiados de tal manera que el elemento más antiguo estará al principio de la memoria física que ha sido destinada para el array. Existen 3 variantes de la función.

Llamada según la posición de inicio y el número de elementos requeridos

```
int  CopyOpen(
   string            symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES   timeframe,        // período
   int               start_pos,        // posición de inicio
   int               count,            // cantidad de datos a copiar
   double            open_array[]      // array para copiar los precios de apertura
   );
```

Llamada según la fecha de inicio y el número de elementos requeridos

```
int  CopyOpen(
   string            symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES   timeframe,        // período
   datetime          start_time,       // fecha y hora de inicio
   int               count,            // cantidad de datos a copiar
   double            open_array[]      // array para copiar los precios de apertura
   );
```

Llamada según la fecha de inicio y finalización del intervalo de tiempo requerido

© 2000-2025, MetaQuotes Ltd.

```
int  CopyOpen(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   datetime         start_time,       // fecha y hora de inicio
   datetime         stop_time,        // fecha y hora de finalización
   double           open_array[]      // array para copiar los precios de apertura
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_pos

[in] Posición del primer elemento a copiar.

count

[in] Cantidad de elementos a copiar.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

open_array[]

[out] Array del tipo double.

Valor devuelto

Cantidad de elementos copiados del array, o -1 en caso del error.

Nota

Si el intervalo de datos solicitados se encuentra totalmente fuera del rango de datos disponibles del servidor, la función devuelve -1. Si los datos solicitados salen del rango de TERMINAL_MAXBARS (la cantidad máxima de barras en el gráfico), la función tambien devuelve -1.

Al solicitar los datos del indicador, si las series temporales solicitadas todavía no están construidas o hay que bajarlas del servidor, la función devolverá inmediatamente -1. Aunque en este caso, se iniciará el proceso de descarga/construcción.

Cuando se solicitan los datos a un Asesor Experto o un script, se iniciará la descarga desde el servidor si el terminal no dispone de estos datos a nivel local, o se empezará la construcción de la serie temporal necesaria si se puede construir los datos usando el historial local y ellos todavía no están listos. La función devolverá aquella cantidad de datos que estarán listos para el momento de vencimiento de timeout, pero el historial seguirá cargándose y con la siguiente solicitud del mismo tipo la función devolverá más datos.

Cuando se solicitan los datos por la fecha inicial y el número de elementos requeridos, sólo se devuelven los datos cuya fecha es menor (anterior) o igual a la especificada. Este intervalo se

© 2000-2025, MetaQuotes Ltd.

establece y se cuenta con la precisión de segundos. Es decir, la fecha de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre es igual o menor que la especificada.

Cuando se solicitan los datos de un período de fechas especificado, se devuelven los datos que entran sólo en este intervalo temporal especificado. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la hora de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre se encuentra en el intervalo especificado.

Por ejemplo, si el día corriente es sabado, al intentar copiar los datos del margen semanal indicando *start_time=Último_Martes* y *stop_time=Último_Viernes*, la función devolverá 0 porque la apertura en un período de tiempo semanal siempre cae en domingo, pero ninguna barra semanal no entra en el período especificado.

Si se necesita obtener el valor que corresponde a una barra corriente no finalizada, se puede usar la primera forma de llamada, indicando *start_pos*=0 y *count*=1.

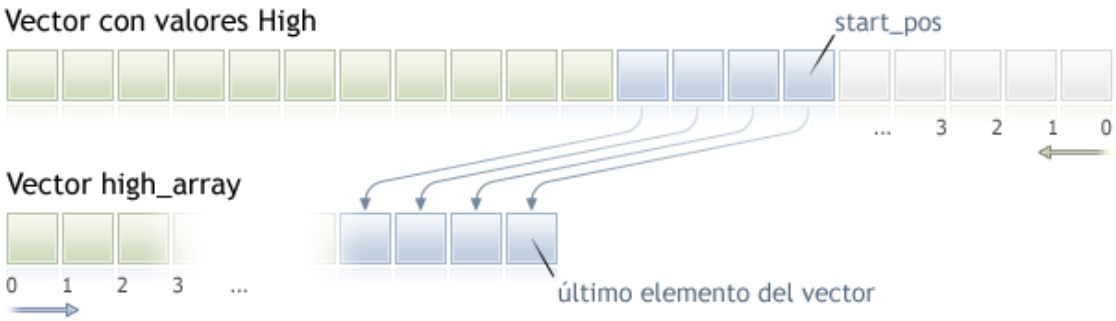
Puede encontrar un ejemplo de la solicitud de datos históricos en la sección Modos de enlace de objetos. En el script de aquella sección se muestra cómo se puede obtener los valores del indicador iFractals sobre las últimas 1000 barras, y cómo luego se puede visualizar en el gráfico los diez últimos fractales arriba y los diez último fractales abajo. Esta técnica puede ser utilizada para todos los indicadores que tienen omisiones de valores y suelen realizarse utilizando los siguientes estilos de construcción:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyHigh

La función recibe en el array high_array datos históricos de los precios máximos de barras para el par especificado de símbolo-período en cantidad especificada. Cabe mencionar que la cuenta de elementos desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente.



Cuando copiamos los datos sin conocer todavía el volumen a copiar, se recomienda usar un array dinámico como array de destino, porque si la cantidad de datos resulta ser menos (o más) de la que cabe en el array, entonces se intenta redistribuirlo de tal manera para que los datos solicitados quepan íntegramente.

Si sabemos la cantidad de datos que tenemos que copiar, con el fin de evitar la asignación excesiva de memoria es mejor hacerlo a un buffer asignado estáticamente.

La propiedad del array de destino no importa, sea as_series=true o sea as_series=false. Los datos van a ser copiados de tal manera que el elemento más antiguo estará al principio de la memoria física que ha sido destinada para el array. Existen 3 variantes de la función.

Llamada según la posición de inicio y el número de elementos requeridos

```
int  CopyHigh(
   string            symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES   timeframe,        // período
   int               start_pos,        // posición de inicio
   int               count,            // cantidad de datos a copiar
   double            high_array[]      // array para copiar los precios máximos
   );
```

Llamada según la fecha de inicio y el número de elementos requeridos

```
int  CopyHigh(
   string            symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES   timeframe,        // período
   datetime          start_time,       // fecha y hora de inicio
   int               count,            // cantidad de datos a copiar
   double            high_array[]      // array para copiar los precios máximos
   );
```

Llamada según la fecha de inicio y finalización del intervalo de tiempo requerido

© 2000-2025, MetaQuotes Ltd.

```
int  CopyHigh(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   datetime         start_time,       // fecha y hora de inicio
   datetime         stop_time,        // fecha y hora de finalización
   double           high_array[]      // array para copiar los precios máximos
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_pos

[in] Posición del primer elemento a copiar.

count

[in] Cantidad de elementos a copiar.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

high_array[]

[out] Array del tipo double.

Valor devuelto

Cantidad de elementos copiados del array, o -1 en caso del error.

Nota

Si el intervalo de datos solicitados se encuentra totalmente fuera del rango de datos disponibles del servidor, la función devuelve -1. Si los datos solicitados salen del rango de TERMINAL_MAXBARS (la cantidad máxima de barras en el gráfico), la función tambien devuelve -1.

Al solicitar los datos del indicador, si las series temporales solicitadas todavía no están construidas o hay que bajarlas del servidor, la función devolverá inmediatamente -1. Aunque en este caso, se iniciará el proceso de descarga/construcción.

Cuando se solicitan los datos a un Asesor Experto o un script, se iniciará la descarga desde el servidor si el terminal no dispone de estos datos a nivel local, o se empezará la construcción de la serie temporal necesaria si se puede construir los datos usando el historial local y ellos todavía no están listos. La función devolverá aquella cantidad de datos que estarán listos para el momento de vencimiento de timeout, pero el historial seguirá cargándose y con la siguiente solicitud del mismo tipo la función devolverá más datos.

Cuando se solicitan los datos por la fecha inicial y el número de elementos requeridos, sólo se devuelven los datos cuya fecha es menor (anterior) o igual a la especificada. Este intervalo se

© 2000-2025, MetaQuotes Ltd.

establece y se cuenta con la precisión de segundos. Es decir, la fecha de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre es igual o menor que la especificada.

Cuando se solicitan los datos de un período de fechas especificado, se devuelven los datos que entran sólo en este intervalo temporal especificado. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la hora de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre se encuentra en el intervalo especificado.

Por ejemplo, si el día corriente es sabado, al intentar copiar los datos del margen semanal indicando *start_time=Último_Martes* y *stop_time=Último_Viernes*, la función devolverá 0 porque la apertura en un período de tiempo semanal siempre cae en domingo, pero ninguna barra semanal no entra en el período especificado.

Si se necesita obtener el valor que corresponde a una barra corriente no finalizada, se puede usar la primera forma de llamada, indicando *start_pos*=0 y *count*=1.

Ejemplo:

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un ejemplo de visualizar los valores High[i] y Low[i]"
#property description "para las barras seleccionadas de una manera aleatoria"

double High[],Low[];
//+------------------------------------------------------------------+
//| Obtenemos Low para el índice especificado de la barra            |
//+------------------------------------------------------------------+
double iLow(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double low=0;
   ArraySetAsSeries(Low,true);
   int copied=CopyLow(symbol,timeframe,0,Bars(symbol,timeframe),Low);
   if(copied>0 && index<copied) low=Low[index];
   return(low);
  }
//+------------------------------------------------------------------+
//| Obtenemos High para el índice especificado de la barra           |
//+------------------------------------------------------------------+
double iHigh(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double high=0;
   ArraySetAsSeries(High,true);
   int copied=CopyHigh(symbol,timeframe,0,Bars(symbol,timeframe),High);
   if(copied>0 && index<copied) high=High[index];
   return(high);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- mostramos en cada tick los valores de High y Low para e la barra con índice,
//--- que equivale al segundo de llegada del tick
   datetime t=TimeCurrent();
   int sec=t%60;
   printf("High[%d] = %G  Low[%d] = %G",
          sec,iHigh(Symbol(),0,sec),
          sec,iLow(Symbol(),0,sec));
  }
```

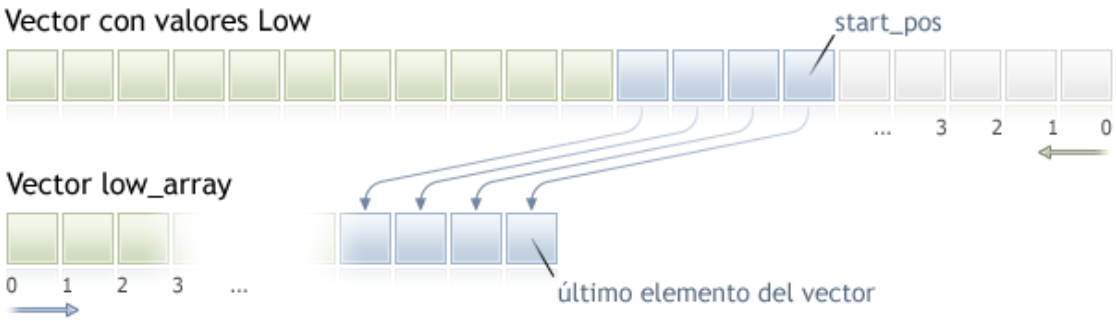
Puede encontrar un ejemplo más detallado de la solicitud de datos históricos en la sección Modos de enlace de objetos. En el script de aquella sección se muestra cómo se puede obtener los valores del indicador iFractals sobre las últimas 1000 barras, y cómo luego se puede visualizar en el gráfico los diez últimos fractales arriba y los diez último fractales abajo. Esta técnica puede ser utilizada para todos los indicadores que tienen omisiones de valores y suelen realizarse utilizando los siguientes estilos de construcción:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyLow

La función recibe en el array low_array datos históricos de los precios mínimos de barras para el par especificado de símbolo-período en cantidad especificada. Cabe mencionar que la cuenta de elementos desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente.



Cuando copiamos los datos sin conocer todavía el volumen a copiar, se recomienda usar un array dinámico como array de destino, porque si la cantidad de datos resulta ser menos (o más) de la que cabe en el array, entonces se intenta redistribuirlo de tal manera para que los datos solicitados quepan íntegramente.

Si sabemos la cantidad de datos que tenemos que copiar, con el fin de evitar la asignación excesiva de memoria es mejor hacerlo a un buffer asignado estáticamente.

La propiedad del array de destino no importa, sea as_series=true o sea as_series=false. Los datos van a ser copiados de tal manera que el elemento más antiguo estará al principio de la memoria física que ha sido destinada para el array. Existen 3 variantes de la función.

Llamada según la posición de inicio y el número de elementos requeridos

```
int  CopyLow(
   string            symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES   timeframe,        // período
   int               start_pos,        // posición de inicio
   int               count,            // cantidad de datos a copiar
   double            low_array[]       // array para copiar los precios mínimos
   );
```

Llamada según la fecha de inicio y el número de elementos requeridos

```
int  CopyLow(
   string            symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES   timeframe,        // período
   datetime          start_time,       // fecha y hora de inicio
   int               count,            // cantidad de datos a copiar
   double            low_array[]       // array para copiar los precios mínimos
   );
```

Llamada según la fecha de inicio y finalización del intervalo de tiempo requerido

© 2000-2025, MetaQuotes Ltd.

```
int  CopyLow(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   datetime         start_time,       // fecha y hora de inicio
   datetime         stop_time,        // fecha y hora de finalización
   double           low_array[]       // array para copiar los precios mínimos
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_pos

[in] Posición del primer elemento a copiar.

count

[in] Cantidad de elementos a copiar.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

low_array[]

[out] Array del tipo double.

Valor devuelto

Cantidad de elementos copiados del array, o -1 en caso del error.

Nota

Si el intervalo de datos solicitados se encuentra totalmente fuera del rango de datos disponibles del servidor, la función devuelve -1. Si los datos solicitados salen del rango de TERMINAL_MAXBARS (la cantidad máxima de barras en el gráfico), la función tambien devuelve -1.

Al solicitar los datos del indicador, si las series temporales solicitadas todavía no están construidas o hay que bajarlas del servidor, la función devolverá inmediatamente -1. Aunque en este caso, se iniciará el proceso de descarga/construcción.

Cuando se solicitan los datos a un Asesor Experto o un script, se iniciará la descarga desde el servidor si el terminal no dispone de estos datos a nivel local, o se empezará la construcción de la serie temporal necesaria si se puede construir los datos usando el historial local y ellos todavía no están listos. La función devolverá aquella cantidad de datos que estarán listos para el momento de vencimiento de timeout, pero el historial seguirá cargándose y con la siguiente solicitud del mismo tipo la función devolverá más datos.

Cuando se solicitan los datos por la fecha inicial y el número de elementos requeridos, sólo se devuelven los datos cuya fecha es menor (anterior) o igual a la especificada. Este intervalo se

© 2000-2025, MetaQuotes Ltd.

establece y se cuenta con la precisión de segundos. Es decir, la fecha de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre es igual o menor que la especificada.

Cuando se solicitan los datos de un período de fechas especificado, se devuelven los datos que entran sólo en este intervalo temporal especificado. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la hora de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre se encuentra en el intervalo especificado.

Por ejemplo, si el día corriente es sabado, al intentar copiar los datos del margen semanal indicando *start_time=Último_Martes* y *stop_time=Último_Viernes*, la función devolverá 0 porque la apertura en un período de tiempo semanal siempre cae en domingo, pero ninguna barra semanal no entra en el período especificado.

Si se necesita obtener el valor que corresponde a una barra corriente no finalizada, se puede usar la primera forma de llamada, indicando *start_pos*=0 y *count*=1.

Puede encontrar un ejemplo de la solicitud de datos históricos en la sección Modos de enlace de objetos. En el script de aquella sección se muestra cómo se puede obtener los valores del indicador iFractals sobre las últimas 1000 barras, y cómo luego se puede visualizar en el gráfico los diez últimos fractales arriba y los diez último fractales abajo. Esta técnica puede ser utilizada para todos los indicadores que tienen omisiones de valores y suelen realizarse utilizando los siguientes estilos de construcción:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

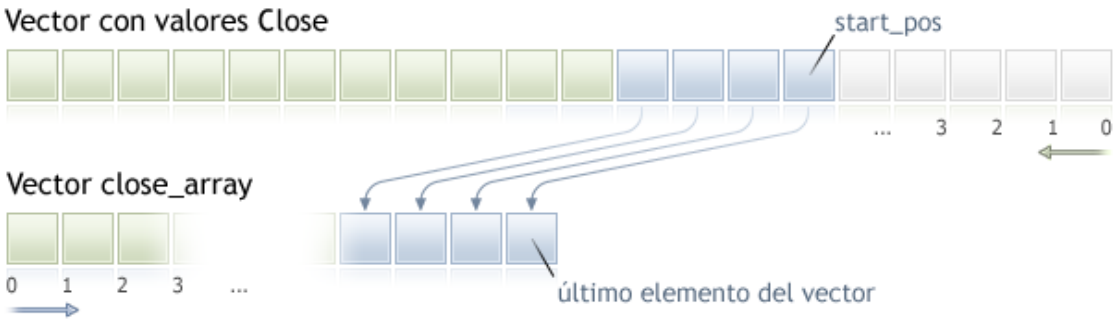
Véase también

CopyHigh

© 2000-2025, MetaQuotes Ltd.

CopyClose

La función recibe en el array close_array datos históricos de los precios del cierre de barras para el par especificado de símbolo-período en cantidad especificada. Cabe mencionar que la cuenta de elementos desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente.



Cuando copiamos los datos sin conocer todavía el volumen a copiar, se recomienda usar un array dinámico como array de destino, porque si la cantidad de datos resulta ser menos (o más) de la que cabe en el array, entonces se intenta redistribuirlo de tal manera para que los datos solicitados quepan íntegramente.

Si sabemos la cantidad de datos que tenemos que copiar, con el fin de evitar la asignación excesiva de memoria es mejor hacerlo a un buffer asignado estáticamente.

La propiedad del array de destino no importa, sea as_series=true o sea as_series=false. Los datos van a ser copiados de tal manera que el elemento más antiguo estará al principio de la memoria física que ha sido destinada para el array. Existen 3 variantes de la función.

Llamada según la posición de inicio y el número de elementos requeridos

```
int  CopyClose(
   string           symbol_name,       // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,         // período
   int              start_pos,         // posición de inicio
   int              count,             // cantidad de datos a copiar
   double           close_array[]      // array para copiar los precios de cierre
   );
```

Llamada según la fecha de inicio y el número de elementos requeridos

```
int  CopyClose(
   string           symbol_name,       // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // fecha y hora de inicio
   int              count,             // cantidad de datos a copiar
   double           close_array[]      // array para copiar los precios de cierre
   );
```

Llamada según la fecha de inicio y finalización del intervalo de tiempo requerido

© 2000-2025, MetaQuotes Ltd.

```
int  CopyClose(
   string           symbol_name,       // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // fecha y hora de inicio
   datetime         stop_time,         // fecha y hora de finalización
   double           close_array[]      // array para copiar los precios de cierre
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_pos

[in] Posición del primer elemento a copiar.

count

[in] Cantidad de elementos a copiar.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

close_array[]

[out] Array del tipo double.

Valor devuelto

Cantidad de elementos copiados del array, o -1 en caso del error.

Nota

Si el intervalo de datos solicitados se encuentra totalmente fuera del rango de datos disponibles del servidor, la función devuelve -1. Si los datos solicitados salen del rango de TERMINAL_MAXBARS (la cantidad máxima de barras en el gráfico), la función tambien devuelve -1.

Al solicitar los datos del indicador, si las series temporales solicitadas todavía no están construidas o hay que bajarlas del servidor, la función devolverá inmediatamente -1. Aunque en este caso, se iniciará el proceso de descarga/construcción.

Cuando se solicitan los datos a un Asesor Experto o un script, se iniciará la descarga desde el servidor si el terminal no dispone de estos datos a nivel local, o se empezará la construcción de la serie temporal necesaria si se puede construir los datos usando el historial local y ellos todavía no están listos. La función devolverá aquella cantidad de datos que estarán listos para el momento de vencimiento de timeout, pero el historial seguirá cargándose y con la siguiente solicitud del mismo tipo la función devolverá más datos.

Cuando se solicitan los datos por la fecha inicial y el número de elementos requeridos, sólo se devuelven los datos cuya fecha es menor (anterior) o igual a la especificada. Este intervalo se

© 2000-2025, MetaQuotes Ltd.

establece y se cuenta con la precisión de segundos. Es decir, la fecha de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre es igual o menor que la especificada.

Cuando se solicitan los datos de un período de fechas especificado, se devuelven los datos que entran sólo en este intervalo temporal especificado. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la hora de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre se encuentra en el intervalo especificado.

Por ejemplo, si el día corriente es sabado, al intentar copiar los datos del margen semanal indicando *start_time=Último_Martes* y *stop_time=Último_Viernes*, la función devolverá 0 porque la apertura en un período de tiempo semanal siempre cae en domingo, pero ninguna barra semanal no entra en el período especificado.

Si se necesita obtener el valor que corresponde a una barra corriente no finalizada, se puede usar la primera forma de llamada, indicando *start_pos*=0 y *count*=1.

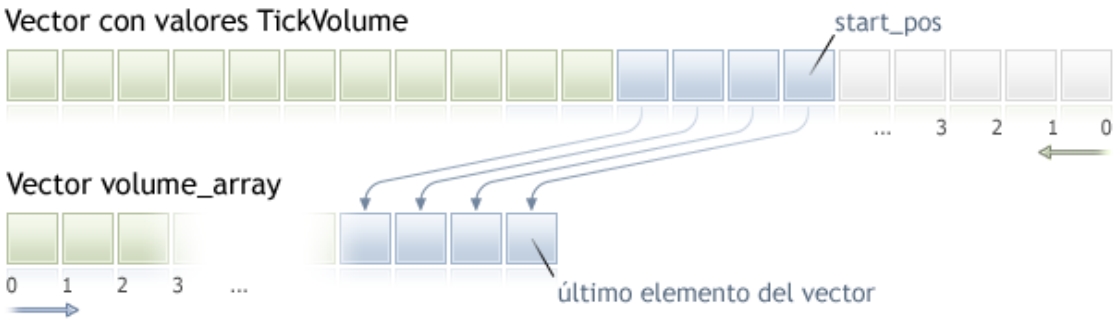
Puede encontrar un ejemplo de la solicitud de datos históricos en la sección Modos de enlace de objetos. En el script de aquella sección se muestra cómo se puede obtener los valores del indicador iFractals sobre las últimas 1000 barras, y cómo luego se puede visualizar en el gráfico los diez últimos fractales arriba y los diez último fractales abajo. Esta técnica puede ser utilizada para todos los indicadores que tienen omisiones de valores y suelen realizarse utilizando los siguientes estilos de construcción:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyTickVolume

La función recibe en el array volume_array datos históricos de los volúmenes de tick para el par especificado de símbolo-período en cantidad especificada. Cabe mencionar que la cuenta de elementos desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente.



Cuando copiamos los datos sin conocer todavía el volumen a copiar, se recomienda usar un array dinámico como array de destino, porque si la cantidad de datos resulta ser menos (o más) de la que cabe en el array, entonces se intenta redistribuirlo de tal manera para que los datos solicitados quepan íntegramente.

Si sabemos la cantidad de datos que tenemos que copiar, con el fin de evitar la asignación excesiva de memoria es mejor hacerlo a un buffer asignado estáticamente.

La propiedad del array de destino no importa, sea as_series=true o sea as_series=false. Los datos van a ser copiados de tal manera que el elemento más antiguo estará al principio de la memoria física que ha sido destinada para el array. Existen 3 variantes de la función.

Llamada según la posición de inicio y el número de elementos requeridos

```
int  CopyTickVolume(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   int              start_pos,        // posición de inicio
   int              count,            // cantidad de datos a copiar
   long             volume_array[]    // array para copiar los volúmenes de tick
   );
```

Llamada según la fecha de inicio y el número de elementos requeridos

```
int  CopyTickVolume(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   datetime         start_time,       // fecha y hora de inicio
   int              count,            // cantidad de datos a copiar
   long             volume_array[]    // array para copiar los volúmenes de tick
   );
```

Llamada según la fecha de inicio y finalización del intervalo de tiempo requerido

© 2000-2025, MetaQuotes Ltd.

```
int  CopyTickVolume(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   datetime         start_time,       // fecha y hora de inicio
   datetime         stop_time,        // fecha y hora de finalización
   long             volume_array[]    // array para copiar los volúmenes de tick
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_pos

[in] Posición del primer elemento a copiar.

count

[in] Cantidad de elementos a copiar.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

volume_array[]

[out] Array del tipo long.

Valor devuelto

Cantidad de elementos copiados del array, o -1 en caso del error.

Nota

Si el intervalo de datos solicitados se encuentra totalmente fuera del rango de datos disponibles del servidor, la función devuelve -1. Si los datos solicitados salen del rango de TERMINAL_MAXBARS (la cantidad máxima de barras en el gráfico), la función tambien devuelve -1.

Al solicitar los datos del indicador, si las series temporales solicitadas todavía no están construidas o hay que bajarlas del servidor, la función devolverá inmediatamente -1. Aunque en este caso, se iniciará el proceso de descarga/construcción.

Cuando se solicitan los datos a un Asesor Experto o un script, se iniciará la descarga desde el servidor si el terminal no dispone de estos datos a nivel local, o se empezará la construcción de la serie temporal necesaria si se puede construir los datos usando el historial local y ellos todavía no están listos. La función devolverá aquella cantidad de datos que estarán listos para el momento de vencimiento de timeout, pero el historial seguirá cargándose y con la siguiente solicitud del mismo tipo la función devolverá más datos.

Cuando se solicitan los datos por la fecha inicial y el número de elementos requeridos, sólo se devuelven los datos cuya fecha es menor (anterior) o igual a la especificada. Este intervalo se

© 2000-2025, MetaQuotes Ltd.

establece y se cuenta con la precisión de segundos. Es decir, la fecha de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre es igual o menor que la especificada.

Cuando se solicitan los datos de un período de fechas especificado, se devuelven los datos que entran sólo en este intervalo temporal especificado. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la hora de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre se encuentra en el intervalo especificado.

Por ejemplo, si el día corriente es sabado, al intentar copiar los datos del margen semanal indicando *start_time=Último_Martes* y *stop_time=Último_Viernes*, la función devolverá 0 porque la apertura en un período de tiempo semanal siempre cae en domingo, pero ninguna barra semanal no entra en el período especificado.

Si se necesita obtener el valor que corresponde a una barra corriente no finalizada, se puede usar la primera forma de llamada, indicando *start_pos*=0 y *count*=1.

Ejemplo:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot TickVolume
#property indicator_label1  "TickVolume"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  C'143,188,139'
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      bars=3000;
//--- indicator buffers
double         TickVolumeBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

© 2000-2025, MetaQuotes Ltd.

```
                  const double &low[],
                  const double &close[],
                  const long &tick_volume[],
                  const long &volume[],
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      long timeseries[];
      ArraySetAsSeries(timeseries,true);
      int prices=CopyTickVolume(Symbol(),0,0,bars,timeseries);
      for(int i=0;i<rates_total-prices;i++) TickVolumeBuffer[i]=0.0;
      for(int i=0;i<prices;i++) TickVolumeBuffer[rates_total-1-i]=timeseries[prices-1-
      Print("Recibida la siguiente cantidad de valores históricos TickVolume: "+prices
     }
   else
     {
      long timeseries[];
      int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
      TickVolumeBuffer[rates_total-1]=timeseries[0];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

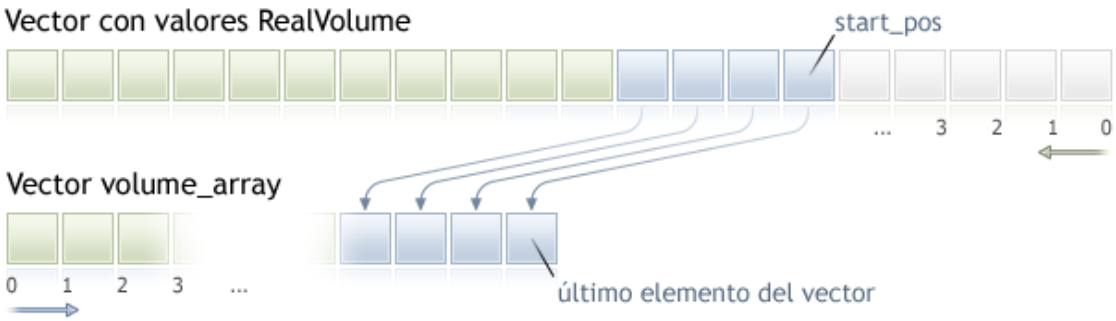
Puede encontrar un ejemplo más detallado de la solicitud de datos históricos en la sección Modos de enlace de objetos. En el script de aquella sección se muestra cómo se puede obtener los valores del indicador iFractals sobre las últimas 1000 barras, y cómo luego se puede visualizar en el gráfico los diez últimos fractales arriba y los diez último fractales abajo. Esta técnica puede ser utilizada para todos los indicadores que tienen omisiones de valores y suelen realizarse utilizando los siguientes estilos de construcción:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyRealVolume

La función recibe en el array volume_array datos históricos de los volúmenes comerciales para el par especificado de símbolo-período en cantidad especificada. Cabe mencionar que la cuenta de elementos desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente.



Cuando copiamos los datos sin conocer todavía el volumen a copiar, se recomienda usar un array dinámico como array de destino, porque si la cantidad de datos resulta ser menos (o más) de la que cabe en el array, entonces se intenta redistribuirlo de tal manera para que los datos solicitados quepan íntegramente.

Si sabemos la cantidad de datos que tenemos que copiar, con el fin de evitar la asignación excesiva de memoria es mejor hacerlo a un buffer asignado estáticamente.

La propiedad del array de destino no importa, sea as_series=true o sea as_series=false. Los datos van a ser copiados de tal manera que el elemento más antiguo estará al principio de la memoria física que ha sido destinada para el array. Existen 3 variantes de la función.

Llamada según la posición de inicio y el número de elementos requeridos

```
int  CopyRealVolume(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   int              start_pos,        // posición de inicio
   int              count,            // cantidad de datos a copiar
   long             volume_array[]    // array para copiar los volúmenes
   );
```

Llamada según la fecha de inicio y el número de elementos requeridos

```
int  CopyRealVolume(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   datetime         start_time,       // fecha y hora de inicio
   int              count,            // cantidad de datos a copiar
   long             volume_array[]    // array para copiar los volúmenes
   );
```

Llamada según la fecha de inicio y finalización del intervalo de tiempo requerido

© 2000-2025, MetaQuotes Ltd.

```
int  CopyRealVolume(
   string           symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES  timeframe,        // período
   datetime         start_time,       // fecha y hora de inicio
   datetime         stop_time,        // fecha y hora de finalización
   long             volume_array[]    // array para copiar los volúmenes
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_pos

[in] Posición del primer elemento a copiar.

count

[in] Cantidad de elementos a copiar.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

volume_array[]

[out] Array del tipo long.

Valor devuelto

Cantidad de elementos copiados del array, o -1 en caso del error.

Nota

Si el intervalo de datos solicitados se encuentra totalmente fuera del rango de datos disponibles del servidor, la función devuelve -1. Si los datos solicitados salen del rango de TERMINAL_MAXBARS (la cantidad máxima de barras en el gráfico), la función tambien devuelve -1.

Al solicitar los datos del indicador, si las series temporales solicitadas todavía no están construidas o hay que bajarlas del servidor, la función devolverá inmediatamente -1. Aunque en este caso, se iniciará el proceso de descarga/construcción.

Cuando se solicitan los datos a un Asesor Experto o un script, se iniciará la descarga desde el servidor si el terminal no dispone de estos datos a nivel local, o se empezará la construcción de la serie temporal necesaria si se puede construir los datos usando el historial local y ellos todavía no están listos. La función devolverá aquella cantidad de datos que estarán listos para el momento de vencimiento de timeout, pero el historial seguirá cargándose y con la siguiente solicitud del mismo tipo la función devolverá más datos.

Cuando se solicitan los datos por la fecha inicial y el número de elementos requeridos, sólo se devuelven los datos cuya fecha es menor (anterior) o igual a la especificada. Este intervalo se

© 2000-2025, MetaQuotes Ltd.

establece y se cuenta con la precisión de segundos. Es decir, la fecha de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre es igual o menor que la especificada.

Cuando se solicitan los datos de un período de fechas especificado, se devuelven los datos que entran sólo en este intervalo temporal especificado. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la hora de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre se encuentra en el intervalo especificado.

Por ejemplo, si el día corriente es sabado, al intentar copiar los datos del margen semanal indicando *start_time=Último_Martes* y *stop_time=Último_Viernes*, la función devolverá 0 porque la apertura en un período de tiempo semanal siempre cae en domingo, pero ninguna barra semanal no entra en el período especificado.

Si se necesita obtener el valor que corresponde a una barra corriente no finalizada, se puede usar la primera forma de llamada, indicando *start_pos*=0 y *count*=1.

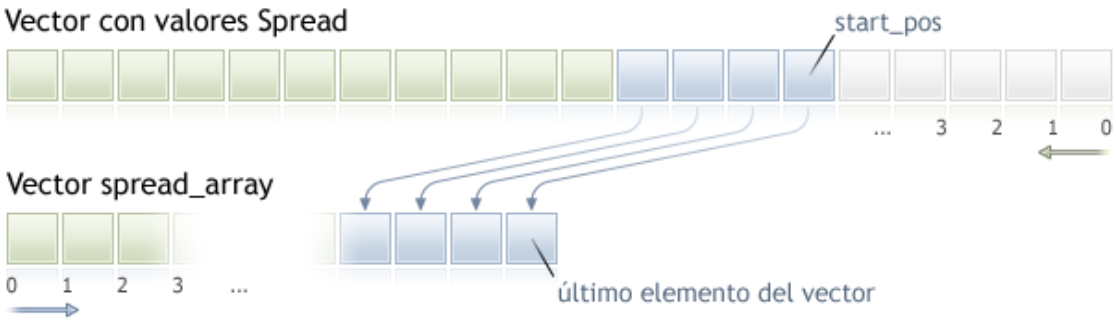
Puede encontrar un ejemplo de la solicitud de datos históricos en la sección Modos de enlace de objetos. En el script de aquella sección se muestra cómo se puede obtener los valores del indicador iFractals sobre las últimas 1000 barras, y cómo luego se puede visualizar en el gráfico los diez últimos fractales arriba y los diez último fractales abajo. Esta técnica puede ser utilizada para todos los indicadores que tienen omisiones de valores y suelen realizarse utilizando los siguientes estilos de construcción:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopySpread

La función recibe en el array spread_array datos históricos de los spreads para el par especificado de símbolo-período en cantidad especificada. Cabe mencionar que la cuenta de elementos desde la posición de inicio se realiza del presente al pasado, es decir, la posición de inicio igual a 0 significa la barra corriente.



Cuando copiamos los datos sin conocer todavía el volumen a copiar, se recomienda usar un array dinámico como array de destino, porque si la cantidad de datos resulta ser menos (o más) de la que cabe en el array, entonces se intenta redistribuirlo de tal manera para que los datos solicitados quepan íntegramente.

Si sabemos la cantidad de datos que tenemos que copiar, con el fin de evitar la asignación excesiva de memoria es mejor hacerlo a un buffer asignado estáticamente.

La propiedad del array de destino no importa, sea as_series=true o sea as_series=false. Los datos van a ser copiados de tal manera que el elemento más antiguo estará al principio de la memoria física que ha sido destinada para el array. Existen 3 variantes de la función.

Llamada según la posición de inicio y el número de elementos requeridos

```
int  CopySpread(
   string            symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES   timeframe,        // período
   int               start_pos,        // posición de inicio
   int               count,            // cantidad de datos a copiar
   int               spread_array[]    // array para copiar los spreads
   );
```

Llamada según la fecha de inicio y el número de elementos requeridos

```
int  CopySpread(
   string            symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES   timeframe,        // período
   datetime          start_time,       // fecha y hora de inicio
   int               count,            // cantidad de datos a copiar
   int               spread_array[]    // array para copiar los spreads
   );
```

Llamada según la fecha de inicio y finalización del intervalo de tiempo requerido

© 2000-2025, MetaQuotes Ltd.

```
int  CopySpread(
   string            symbol_name,      // nombre del símbolo
   ENUM_TIMEFRAMES   timeframe,        // período
   datetime          start_time,       // fecha y hora de inicio
   datetime          stop_time,        // fecha y hora de finalización
   int               spread_array[]    // array para copiar los spreads
   );
```

Parámetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_pos

[in] Posición del primer elemento a copiar.

count

[in] Cantidad de elementos a copiar.

start_time

[in] Hora de la barra correspondiente al primer elemento.

stop_time

[in] Hora de la barra correspondiente al último elemento.

spread_array[]

[out] Array del tipo int.

Valor devuelto

Cantidad de elementos copiados del array, o -1 en caso del error.

Nota

Si el intervalo de datos solicitados se encuentra totalmente fuera del rango de datos disponibles del servidor, la función devuelve -1. Si los datos solicitados salen del rango de TERMINAL_MAXBARS (la cantidad máxima de barras en el gráfico), la función tambien devuelve -1.

Al solicitar los datos del indicador, si las series temporales solicitadas todavía no están construidas o hay que bajarlas del servidor, la función devolverá inmediatamente -1. Aunque en este caso, se iniciará el proceso de descarga/construcción.

Cuando se solicitan los datos a un Asesor Experto o un script, se iniciará la descarga desde el servidor si el terminal no dispone de estos datos a nivel local, o se empezará la construcción de la serie temporal necesaria si se puede construir los datos usando el historial local y ellos todavía no están listos. La función devolverá aquella cantidad de datos que estarán listos para el momento de vencimiento de timeout, pero el historial seguirá cargándose y con la siguiente solicitud del mismo tipo la función devolverá más datos.

Cuando se solicitan los datos por la fecha inicial y el número de elementos requeridos, sólo se devuelven los datos cuya fecha es menor (anterior) o igual a la especificada. Este intervalo se

© 2000-2025, MetaQuotes Ltd.

establece y se cuenta con la precisión de segundos. Es decir, la fecha de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre es igual o menor que la especificada.

Cuando se solicitan los datos de un período de fechas especificado, se devuelven los datos que entran sólo en este intervalo temporal especificado. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la hora de apertura de cualquier barra para la que se devuelve el valor (volumen, spread, valor en el búfer de indicador, precio Open, High, Low, Close o la hora de apertura Time) siempre se encuentra en el intervalo especificado.

Por ejemplo, si el día corriente es sabado, al intentar copiar los datos del margen semanal indicando *start_time=Último_Martes* y *stop_time=Último_Viernes*, la función devolverá 0 porque la apertura en un período de tiempo semanal siempre cae en domingo, pero ninguna barra semanal no entra en el período especificado.

Si se necesita obtener el valor que corresponde a una barra corriente no finalizada, se puede usar la primera forma de llamada, indicando *start_pos*=0 y *count*=1.

Ejemplo:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Spread
#property indicator_label1  "Spread"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      bars=3000;
//--- indicator buffers
double         SpreadBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

© 2000-2025, MetaQuotes Ltd.

```
                  const double &low[],
                  const double &close[],
                  const long &tick_volume[],
                  const long &volume[],
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      int spread_int[];
      ArraySetAsSeries(spread_int,true);
      int spreads=CopySpread(Symbol(),0,0,bars,spread_int);
      Print("Recibida la siguiente cantidad de valores históricos del spread: ",spread
      for (int i=0;i<spreads;i++)
      {
      SpreadBuffer[rates_total-1-i]=spread_int[i];
      if(i<=30) Print("spread["+i+"] = ",spread_int[i]);
      }
     }
   else
     {
      double Ask,Bid;
      Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
      Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
      Comment("Ask = ",Ask,"  Bid = ",Bid);
      SpreadBuffer[rates_total-1]=(Ask-Bid)/Point();
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Puede encontrar un ejemplo más detallado de la solicitud de datos históricos en la sección Modos de enlace de objetos. En el script de aquella sección se muestra cómo se puede obtener los valores del indicador iFractals sobre las últimas 1000 barras, y cómo luego se puede visualizar en el gráfico los diez últimos fractales arriba y los diez último fractales abajo. Esta técnica puede ser utilizada para todos los indicadores que tienen omisiones de valores y suelen realizarse utilizando los siguientes estilos de construcción:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyTicks

La función recibe en la matriz ticks_array los ticks en formato MqlTick, además, la indexación se tiene lugar del pasado al presente, es decir, el tick con el índice 0 es el más antiguo en la matriz. Para analizar un tick es necesario comprobar el campo *flags*, que nos informa sobre lo que se ha cambiado precisamente en este tick.

```
int  CopyTicks(
   string           symbol_name,           // nombre del símbolo
   MqlTick&         ticks_array[],         // matriz para recibir ticks
   uint             flags=COPY_TICKS_ALL,  // bandera que define el tipo de ticks reci
   ulong            from=0,                // fecha a partir de la cual se solicitan l
   uint             count=0                // número de ticks que se deben recibir
   );
```

Parámetros

symbol_name

[in] Símbolo.

ticks_array

[out] Matriz del tipo MqlTick para recibir los ticks.

flags

[in] bandera que define el tipo de los ticks solicitados. COPY_TICKS_INFO - ticks llamados por los cambios de Bid y/o Ask, COPY_TICKS_TRADE - ticks con los cambios de Last y Volume, COPY_TICKS_ALL - todos los ticks. Al realizarse cualquier tipo de solicitud, en los campos restantes de la estructura MqlTick se acaban de escribir los valores del tick anterior.

from

[in] Fecha a partir de la cual se solicitan los ticks. Se indica en milisegundos desde el 01.01.1970. Si el parámetro *from*=0, entonces se dan los últimos *count* ticks.

count

[in] Número de ticks solicitados. Si los parámetros from y count no se indican, entonces en la matriz ticks_array[] se grabarán todos los últimos ticks disponibles, pero no más de 2000.

Valor devuelto

Número de ticks copiados, o bien -1 en caso de error.

Observación

La función CopyTicks() permite solicitar y analizar todos los ticks entrantes. La primera llamada de CopyTicks() inicia la sincronización de la base de ticks que se guarda en el disco duro de este símbolo. Si los ticks en la base local son insuficientes, entonces los ticks que faltan se cargarán de forma automática desde el servidor comercial. En este caso, los ticks serán sincronizados desde la fecha ***from***, indicada en CopyTicks(), hasta el momento actual. Después de ello, todos los ticks entrantes de dicho símbolo llegarán a la base de ticks y la mantendrán en su actual estado sincronizado.

© 2000-2025, MetaQuotes Ltd.

Si los parámetros *from* y *count* no han sido indicados, entonces en la matriz *ticks_array[]* se grabarán todos los ticks disponibles, pero no más de 2000. El parámetro *flags* permite definir el tipo de los ticks necesarios.

COPY_TICKS_INFO – se dan los ticks en los que hay cambios del precio Bid y/o Ask. Pero además, también se rellenarán los datos del resto de los campos, por ejemplo, si solo ha cambiado el precio Bid, en los campos *ask* y *volume* se grabarán los últimos valores conocidos. Para saber exactamente qué ha sido lo que ha cambiado, es necesario analizar el campo *flags*, que tendrá el valor TICK_FLAG_BID y/o TICK_FLAG_ASK. Si el tick tiene valores cero de los precios Bid y Ask, y las banderas muestran que los precios han cambiado (*flags*=TICK_FLAG_BID|TICK_FLAG_ASK), entonces esto nos indica que se ha vaciado la profundidad de mercado. En otras palabras, en este momento no hay solicitudes de compra o venta.

COPY_TICKS_TRADE – se dan los ticks en los que hay cambios del último precio de la operación y del volumen. Pero además, también se rellenarán los datos del resto de los campos, es decir, en los campos Bid y Ask se grabarán los últimos resultados conocidos. Para saber exactamente qué ha sido lo que ha cambiado, es necesario analizar el campo *flags*, que tendrá el valor TICK_FLAG_LAST y TICK_FLAG_VOLUME.

COPY_TICKS_ALL – se dan todos los ticks en los que hay aunque sea un cambio. Además, los campos no modificados también se rellenan con los últimos valores conocidos.

La llamada de CopyTicks() con la bandera COPY_TICKS_ALL proporciona de golpe todos los ticks del diapasón solicitado, al tiempo que la llamada en otros modos exige de cierto tiempo para el pre-procesado y la selección de los ticks, y por eso no da una ventaja sustancial en cuanto a velocidad de ejecución.

Al solicitar los ticks (no importa si se trata de COPY_TICKS_INFO o de COPY_TICKS_TRADE), en cada tick se contiene información de precio completa en el momento del tick (*bid, ask, last* y *volume*). Esto se ha hecho para que sea más cómodo analizar las circunstancias comerciales en el momento de cada tick, y que no resulte necesario solicitar cada vez la historia profunda de ticks y buscar en ella los valores de otros campos.

En los indicadores, la función CopyTicks() retorna el resultado de inmediato: Al llamar CopyTick() desde el indicador se retornarán de inmediato los ticks disponibles del símbolo, y se iniciará la sincronización de la base de ticks, si los datos han sido insuficientes. Todos los indicadores en un símbolo funcionan en un flujo común, por eso el indicador no tiene derecho a esperar la finalización de la sincronización. Después de finalizar la sincronización, con la siguiente llamada, CopyTicks() retornará todos los ticks solicitados. La función OnCalculate() en los indicadores se llama después de que llegue cada tick.

En los expertos y scripts, la función CopyTicks() puede esperar el resultado hasta 45 segundos: A diferencia del indicador, cada experto y script funciona en su propio flujo, y por eso puede esperar la finalización de la sincronización hasta 45 segundos. Si durante este tiempo los ticks aún no se han sincronizando en el número necesario, entonces CopyTicks() retornará por timeout solo los ticks disponibles, y la sincronización continuará. La función OnTick() en los expertos no se constituye como procesador de cada tick, sino que notifica al experto sobre los cambios en el mercado. Los cambios pueden darse en paquetes: al terminal pueden llegar varios ticks al mismo tiempo, pero la función OnTick() será llamada solo una vez para notificar al experto sobre el último estado del mercado.

Velocidad de entrega: el terminal guarda de cada símbolo los 4096 últimos ticks en el caché para el acceso rápido (para los símbolos con la profundidad de mercado iniciada, serán 65536 ticks), las solicitudes de estos datos son las que más rápidamente se ejecutan. Al solicitar los ticks de la

© 2000-2025, MetaQuotes Ltd.

sesión comercial actual más allá de los límites del caché, CopyTicks() recurre ya a los ticks que se guardan en la memoria del terminal, estas solicitudes necesitan más tiempo para ejecutarse. Las más lentas son las solicitudes de los ticks de otros días, ya que, en este caso, los datos se leen ya desde el disco.

Ejemplo:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- Requesting 100 million ticks to be sure we receive the entire tick history
input int      getticks=100000000; // The number of required ticks
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int     attempts=0;     // Count of attempts
   bool    success=false;  // The flag of a successful copying of ticks
   MqlTick tick_array[];   // Tick receiving array
   MqlTick lasttick;       // To receive last tick data
   SymbolInfoTick(_Symbol,lasttick);
//--- Make 3 attempts to receive ticks
   while(attempts<3)
     {
      //--- Measuring start time before receiving the ticks
      uint start=GetTickCount();
//--- Requesting the tick history since 1970.01.01 00:00.001 (parameter from=1 ms)
      int received=CopyTicks(_Symbol,tick_array,COPY_TICKS_ALL,1,getticks);
      if(received!=-1)
        {
         //--- Showing information about the number of ticks and spent time
         PrintFormat("%s: received %d ticks in %d ms",_Symbol,received,GetTickCount()-
         //--- If the tick history is synchronized, the error code is equal to zero
         if(GetLastError()==0)
           {
            success=true;
            break;
           }
         else
            PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d
            _Symbol,received,GetTickCount()-start,_LastError);
        }
      //--- Counting attempts
      attempts++;
      //--- A one-second pause to wait for the end of synchronization of the tick data
      Sleep(1000);
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//--- Receiving the requested ticks from the beginning of the tick history failed in t
   if(!success)
     {
      PrintFormat("Error! Failed to receive %d ticks of %s in three attempts",getticks
      return;
     }
   int ticks=ArraySize(tick_array);
//--- Showing the time of the first tick in the array
   datetime firstticktime=tick_array[ticks-1].time;
   PrintFormat("Last tick time = %s.%03I64u",
               TimeToString(firstticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar
//--- выведем время последнего тика в массиве
   datetime lastticktime=tick_array[0].time;
   PrintFormat("First tick time = %s.%03I64u",
               TimeToString(lastticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar

//---
   MqlDateTime today;
   datetime current_time=TimeCurrent();
   TimeToStruct(current_time,today);
   PrintFormat("current_time=%s",TimeToString(current_time));
   today.hour=0;
   today.min=0;
   today.sec=0;
   datetime startday=StructToTime(today);
   datetime endday=startday+24*60*60;
   if((ticks=CopyTicksRange(_Symbol,tick_array,COPY_TICKS_ALL,startday*1000,endday*100
     {
      PrintFormat("CopyTicksRange(%s,tick_array,COPY_TICKS_ALL,%s,%s) failed, error %d
                  _Symbol,TimeToString(startday),TimeToString(endday),GetLastError());
      return;
     }
   ticks=MathMax(100,ticks);
//--- Showing the first 100 ticks of the last day
   int counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         counter++;
         PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
        }
     }
//--- Showing the first 100 deals of the last day
   counter=0;
   for(int i=0;i<ticks;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         if(((tick_array[i].flags&TICK_FLAG_BUY)==TICK_FLAG_BUY) || ((tick_array[i].fl
           {
            counter++;
            PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
           }
        }
     }
  }
//+------------------------------------------------------------------+
//| Returns the string description of a tick                         |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03d ",
                            TimeToString(tick.time),tick.time_msc%1000);
//--- Checking flags
   bool buy_tick=((tick.flags&TICK_FLAG_BUY)==TICK_FLAG_BUY);
   bool sell_tick=((tick.flags&TICK_FLAG_SELL)==TICK_FLAG_SELL);
   bool ask_tick=((tick.flags&TICK_FLAG_ASK)==TICK_FLAG_ASK);
   bool bid_tick=((tick.flags&TICK_FLAG_BID)==TICK_FLAG_BID);
   bool last_tick=((tick.flags&TICK_FLAG_LAST)==TICK_FLAG_LAST);
   bool volume_tick=((tick.flags&TICK_FLAG_VOLUME)==TICK_FLAG_VOLUME);
//--- Checking trading flags in a tick first
   if(buy_tick || sell_tick)
     {
      //--- Forming an output for the trading tick
      desc=desc+(buy_tick?StringFormat("Buy Tick: Last=%G Volume=%d ",tick.last,tick.v
      desc=desc+(sell_tick?StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last,tick
      desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
      desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
      desc=desc+"(Trade tick)";
     }
   else
     {
      //--- Form a different output for an info tick
      desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
      desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
      desc=desc+(last_tick?StringFormat("Last=%G ",tick.last):"");
      desc=desc+(volume_tick?StringFormat("Volume=%d ",tick.volume):"");
      desc=desc+"(Info tick)";
     }
//--- Returning tick description
   return desc;
  }
//+------------------------------------------------------------------+
/* Example of the output
```

© 2000-2025, MetaQuotes Ltd.

```
Si-12.16: received 11048387 ticks in 4937 ms
Last tick time = 2016.09.26 18:32:59.775
First tick time = 2015.06.18 09:45:01.000
1.  2016.09.26 09:45.249 Ask=65370 Bid=65370 (Info tick)
2.  2016.09.26 09:47.420 Ask=65370 Bid=65370 (Info tick)
3.  2016.09.26 09:50.893 Ask=65370 Bid=65370 (Info tick)
4.  2016.09.26 09:51.827 Ask=65370 Bid=65370 (Info tick)
5.  2016.09.26 09:53.810 Ask=65370 Bid=65370 (Info tick)
6.  2016.09.26 09:54.491 Ask=65370 Bid=65370 (Info tick)
7.  2016.09.26 09:55.913 Ask=65370 Bid=65370 (Info tick)
8.  2016.09.26 09:59.350 Ask=65370 Bid=65370 (Info tick)
9.  2016.09.26 09:59.678 Bid=65370 (Info tick)
10. 2016.09.26 10:00.000 Sell Tick: Last=65367 Volume=3 (Trade tick)
11. 2016.09.26 10:00.000 Sell Tick: Last=65335 Volume=45 (Trade tick)
12. 2016.09.26 10:00.000 Sell Tick: Last=65334 Volume=95 (Trade tick)
13. 2016.09.26 10:00.191 Sell Tick: Last=65319 Volume=1 (Trade tick)
14. 2016.09.26 10:00.191 Sell Tick: Last=65317 Volume=1 (Trade tick)
15. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=1 (Trade tick)
16. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=10 (Trade tick)
17. 2016.09.26 10:00.191 Sell Tick: Last=65315 Volume=5 (Trade tick)
18. 2016.09.26 10:00.191 Sell Tick: Last=65313 Volume=3 (Trade tick)
19. 2016.09.26 10:00.191 Sell Tick: Last=65307 Volume=25 (Trade tick)
20. 2016.09.26 10:00.191 Sell Tick: Last=65304 Volume=1 (Trade tick)
21. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=1 (Trade tick)
22. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=10 (Trade tick)
23. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=5 (Trade tick)
24. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=1 (Trade tick)
25. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=6 (Trade tick)
26. 2016.09.26 10:00.191 Sell Tick: Last=65299 Volume=1 (Trade tick)
27. 2016.09.26 10:00.191 Bid=65370 (Info tick)
28. 2016.09.26 10:00.232 Ask=65297 (Info tick)
29. 2016.09.26 10:00.276 Sell Tick: Last=65291 Volume=31 (Trade tick)
30. 2016.09.26 10:00.276 Sell Tick: Last=65290 Volume=1 (Trade tick)
*/
```

Véase también

SymbolInfoTick, Estructura para obtener los precios actuales, OnTick()

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange

La función obtiene en la matriz ticks_array los ticks en el formato MqlTick en el rango de fechas indicado. Además, la indexación se realiza del pasado hacia el presente, es decir, el tick con el número 0 es el más antiguo en la matriz. Para analizar un tick es necesario comprobar el campo *flags*, que informa sobre lo que ha cambiado exactamente.

```
int  CopyTicksRange(
   const string     symbol_name,           // nombre del símbolo
   MqlTick&         ticks_array[],         // matriz para la recepción de ticks
   uint             flags=COPY_TICKS_ALL,  // bandera que define el tipo de ticks reci
   ulong            from_msc=0,            // fecha a partir de la cual se solicitan l
   ulong            to_msc=0               // fecha de la que solicitan los ticks
   );
```

Parámetros

symbol_name

[in] Símbolo.

ticks_array

[out] Matriz estática o dinámica MqlTick para recibir los ticks. Si en la matriz dinámica no caben todos los ticks del intervalo de tiempo solicitado, solo se recibirán tantos ticks como quepan en la matriz. En este caso, además, la función generará el error ERR_HISTORY_SMALL_BUFFER (4407) .

flags

[in] Bandera que define el tipo de ticks solicitados. COPY_TICKS_INFO - ticks llamados por el cambio de Bid y/o Ask, COPY_TICKS_TRADE - ticks con los cambios Last y Volume, COPY_TICKS_ALL - todos los ticks. Con cualquier tipo de solicitud, en los campos restantes de la estructura MqlTick se acabarán de escribir los valores del tick anterior.

from_msc

[in] Fecha a partir de la cual se solicitan los ticks. Se indica en milisegundos desde el 01.01.1970. Si el parámetro *from_msc* no ha sido indicado, se proporcionarán los ticks desde el comienzo de la historia. Se entregan los ticks con la hora >= from_msc.

to_msc

[in] Fecha sobre la que se solicitan los ticks. Se indica en milisegundos desde el 01.01.1970. Se entregan los ticks con la fecha y hora <= to_msc. Si el parámetro *to_msc* no se ha indicado, se proporcionarán todos los ticks hasta el final de la historia.

Valor devuelto

Número de ticks copiados, o bien -1, en caso de error. GetLastError() puede retornar los siguientes errores:

- ERR_HISTORY_TIMEOUT - el tiempo de espera de la sicronización de ticks ha terminado, la función ha entregado todo lo que estaba disponible.
- ERR_HISTORY_SMALL_BUFFER - el búfer estático es demasiado pequeño, se ha entregado todo lo que cabía en la matriz.
- ERR_NOT_ENOUGH_MEMORY - no hay memoria suficiente para recibir la historia del rango indicado en la matriz dinámica de ticks. No se ha logrado asignar el volumen de memoria necesario para la matriz de datos.

© 2000-2025, MetaQuotes Ltd.

Nota

La función CopyTicksRange() se ha pensado para solicitar los ticks de un antiguo rango solicitado, por ejemplo, de un día concreto. Mientras que CopyTicks(), por su parte, permite indicar solo la fecha inicial, por ejemplo, obtener todos los ticks desde el inicio del mes hasta el momento actual.

Vea también

SymbolInfoTick, Estructura para obtener los precios actuales, OnTick, CopyTicks

© 2000-2025, MetaQuotes Ltd.

iBars

Retorna el número de barras en la historia según el símbolo y el periodo correspondientes.

```
int  iBars(
   const string           symbol,          // símbolo
   ENUM_TIMEFRAMES        timeframe        // periodo
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

Valor devuelto

Número de barras en el historial según el símbolo y el periodo correspondiente, pero no superior a lo especificado en la configuración de la plataforma en el parámetro "Max. de barras en la ventana" ("Max bars in chart")

Ejemplo:

```
  Print("Bar count on the 'EURUSD,H1' is ",iBars("EURUSD",PERIOD_H1));
```

Vea también

Bars

© 2000-2025, MetaQuotes Ltd.

iBarShift

Busca la barra según la hora y fecha. La función retorna el índice de la barra en el que entra la hora y fecha especificada.

```
int  iBarShift(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // periodo
   datetime            time,            // hora y fecha
   bool                exact=false      // modo
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. PERIOD_CURRENT designa el periodo del gráfico actual.

time

[in] Valor de la hora y fecha para la búsqueda.

exact=false

[in] El valor retornado, si no se ha encontrado la barra para la hora y fecha indicadas. Con el valor *exact=false iBarShift* retorna el índice de la barra más próxima cuya hora y fecha de apertura sea menor a la indicada (*time_open<time*). Si no se ha encotrado esa barra (no hay historia posterior a la hora y fecha indicadas), la función retorna -1. Si *exact=true*, la barra más cercana no será buscada, y la función iBarShift retornará de inmediato -1.

Valor devuelto

índice de la barra en la que entran la hora y fecha indicadas. Si no existe ninguna barra para la hora y fecha indicadas (hay un "agujero" en la historia), la función retorna -1 o el índice de la barra más próxima (dependiendo del parámetro *exact*).

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- la hora y fecha cae en domingo
   datetime time=D'2002.04.25 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- si no existe la barra de la hora y fecha indicadas, iBarShift retornará el índic
   int bar_index=iBarShift(symbol,tf,time,exact);
//--- comprobamos el código de error después de la llamada de iBarShift()
```

© 2000-2025, MetaQuotes Ltd.

```
   int error=GetLastError();
   if(error!=0)
     {
      PrintFormat("iBarShift(): GetLastError=%d - fecha solicitada %s "+
                  "para %s %s no se ha encontrado en la historia disponible",
                  error,TimeToString(time),symbol,EnumToString(tf));
      return;
     }
//--- la función iBarShift() se ha ejecutado con éxito, mostramos los resultados para
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),
               DayOfWeek(time),bar_index,string(exact));
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("Time of bar #%d is %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//--- requerimos encontrar el índice de la barra para la hora y fecha especificadas, s
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
//--- la función iBarShift() se ha ejecutado con éxtito, mostramos los resultados para
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time)
               ,DayOfWeek(time),bar_index,string(exact));
  }
//+------------------------------------------------------------------+
//| Retornamos el nombre para la semana                              |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(SUNDAY);
      break;
      case 1: day=EnumToString(MONDAY);
      break;
      case 2: day=EnumToString(TUESDAY);
      break;
      case 3: day=EnumToString(WEDNESDAY);
      break;
      case 4: day=EnumToString(THURSDAY);
      break;
      case 5: day=EnumToString(FRIDAY);
      break;
      default:day=EnumToString(SATURDAY);
      break;
     }
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   return day;
  }
//+------------------------------------------------------------------+
/* Resultado de la ejecución
   1. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY): bar index is 64 (exact=false)
   Time of bar #64 is 2018.06.08 23:00  (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):bar index is -1  (exact=true)
*/
```

© 2000-2025, MetaQuotes Ltd.

iClose

Retorna el valor del precio de cierre de la barra (indicada por el parámetro shift) del gráfico correspondiente.

```
double  iClose(
   const string         symbol,          // símbolo
   ENUM_TIMEFRAMES      timeframe,       // periodo
   int                  shift            // desplazamiento
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

shift

[in] Índice del valor obtenido de la serie temporal (desplazamiento respecto a la barra actual en el número indicado de barras hacia atrás).

Valor devuelto

Valor del precio de cierre (indicado con el parámetro shift) del gráfico correspondiente o 0 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Nota

La función siempre retorna los datos reales, para ello, envía en cada llamada una solicitud a las series temporales para el símbolo/periodo especificado. Esto significa que si no hay datos listos en la primera llamada de la función, puede requerir algún tiempo preparar el resultado de la ejecución.

La función no guarda los resultados de llamadas anteriores, no existe memoria caché local para el retorno rápido del valor.

Ejemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2025, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Vea también

CopyClose, CopyRates

© 2000-2025, MetaQuotes Ltd.

iHigh

Retorna el valor del precio máximo de la barra (indicada por el parámetro shift) del gráfico correspondiente.

```
double  iHigh(
   const string         symbol,          // símbolo
   ENUM_TIMEFRAMES      timeframe,       // periodo
   int                  shift            // desplazamiento
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

shift

[in] Índice del valor obtenido de la serie temporal (desplazamiento respecto a la barra actual en el número indicado de barras hacia atrás).

Valor devuelto

Valor del precio máximo de la barra (indicado con el parámetro shift) del gráfico correspondiente o 0 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Nota

La función siempre retorna los datos reales, para ello, envía en cada llamada una solicitud a las series temporales para el símbolo/periodo especificado. Esto significa que si no hay datos listos en la primera llamada de la función, puede requerir algún tiempo preparar el resultado de la ejecución.

La función no guarda los resultados de llamadas anteriores, no existe memoria caché local para el retorno rápido del valor.

Ejemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2025, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Vea también

CopyHigh, CopyRates

© 2000-2025, MetaQuotes Ltd.

iHighest

Retorna el índice del mayor valor encontrado (desplazamiento relativo a la barra actual) del gráfico correspondiente.

```
int  iHighest(
   const string        symbol,              // símbolo
   ENUM_TIMEFRAMES     timeframe,           // periodo
   ENUM_SERIESMODE     type,                // identificador de la serie temporal
   int                 count=WHOLE_ARRAY,   // número de elementos
   int                 start=0              // índice
  );
```

Parámetros

symbol

[in] Símbolo en el que se realizará la búsqueda. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

type

[in] Identificador de la serie temporal en la que se realizará la búsqueda. Puede ser cualquiera de los valores de ENUM_SERIESMODE.

count=WHOLE_ARRAY

[in] El número de elementos de las series temporales (en la dirección que va desde la barra actual hacia el aumento del índice), entre los cuales se debe realizar la búsqueda.

start=0

[in] Índice (desplazamiento con respecto a la barra actual) de la barra inicial desde la que comienza la búsqueda del mayor valor. Los valores negativos se ignoran y se reemplazan con un valor cero.

Valor devuelto

Índice del mayor valor encontrado (desplazamiento relativo a la barra actual) del gráfico correspondiente o -1 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Ejemplo:

```
   double val;
//--- cálculo del valor máximo del precio Close en 20 barras consecutivas
//--- comenzando con el índice 4 y terminando con el índice 23 inclusive en el gráfic
   int val_index=iHighest(NULL,0,MODE_CLOSE,20,4);
   if(val_index!=-1)
      val=High[val_index];
   else
      PrintFormat("Error de llamada de iHighest(). Código de error=%d",GetLastError())
```

© 2000-2025, MetaQuotes Ltd.

iLow

Retorna el valor del precio mínimo de la barra (indicada por el parámetro shift) del gráfico correspondiente.

```
double  iLow(
   const string         symbol,          // símbolo
   ENUM_TIMEFRAMES      timeframe,       // periodo
   int                  shift            // desplazamiento
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

shift

[in] Índice del valor obtenido de la serie temporal (desplazamiento respecto a la barra actual en el número indicado de barras hacia atrás).

Valor devuelto

Valor del precio mínimo de la barra (indicado con el parámetro shift) del gráfico correspondiente o 0 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Nota

La función siempre retorna los datos reales, para ello, envía en cada llamada una solicitud a las series temporales para el símbolo/periodo especificado. Esto significa que si no hay datos listos en la primera llamada de la función, puede requerir algún tiempo preparar el resultado de la ejecución.

La función no guarda los resultados de llamadas anteriores, no existe memoria caché local para el retorno rápido del valor.

Ejemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2025, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Vea también

CopyLow, CopyRates

© 2000-2025, MetaQuotes Ltd.

iLowest

Retorna el índice del menor valor encontrado (desplazamiento relativo a la barra actual) del gráfico correspondiente.

```
int  iLowest(
   const string        symbol,              // símbolo
   ENUM_TIMEFRAMES     timeframe,           // periodo
   ENUM_SERIESMODE     type,                // identificador de la serie temporal
   int                 count=WHOLE_ARRAY,   // número de elementos
   int                 start=0              // índice
  );
```

Parámetros

symbol

[in] Símbolo en el que se realizará la búsqueda. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

type

[in] Identificador de la serie temporal en la que se realizará la búsqueda. Puede ser cualquiera de los valores de ENUM_SERIESMODE.

count=WHOLE_ARRAY

[in] El número de elementos de las series temporales (en la dirección que va desde la barra actual hacia el aumento del índice), entre los cuales se debe realizar la búsqueda.

start=0

[in] Índice (desplazamiento con respecto a la barra actual) de la barra inicial desde la que comienza la búsqueda del menor valor. Los valores negativos se ignoran y se reemplazan con un valor cero.

Valor devuelto

Índice del menor valor encontrado (desplazamiento relativo a la barra actual) del gráfico correspondiente o -1 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Ejemplo:

```
   double val;
//--- búsqueda de la barra con el valor mínimo de volumen real en 15 barras consecutiv
//--- comenzando con el índice 10 y terminando con el índice 24 inclusive en el gráfic
   int val_index=iLowest(NULL,0,MODE_REAL_VOLUME,15,10);
   if(val_index!=-1)
      val=Low[val_index];
   else
      PrintFormat("Error de llamada de iLowest(). Código de error=%d",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

iOpen

Retorna el valor del precio de apertura de la barra (indicada por el parámetro shift) del gráfico correspondiente.

```
double  iOpen(
   const string         symbol,          // símbolo
   ENUM_TIMEFRAMES      timeframe,       // periodo
   int                  shift            // desplazamiento
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

shift

[in] Índice del valor obtenido de la serie temporal (desplazamiento respecto a la barra actual en el número indicado de barras hacia atrás).

Valor devuelto

Valor del precio de apertura (indicado con el parámetro shift) del gráfico correspondiente o 0 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Nota

La función siempre retorna los datos reales, para ello, envía en cada llamada una solicitud a las series temporales para el símbolo/periodo especificado. Esto significa que si no hay datos listos en la primera llamada de la función, puede requerir algún tiempo preparar el resultado de la ejecución.

La función no guarda los resultados de llamadas anteriores, no existe memoria caché local para el retorno rápido del valor.

Ejemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2025, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Vea también

CopyOpen, CopyRates

© 2000-2025, MetaQuotes Ltd.

iTime

Retorna el valor del tiempo de apertura de la barra (indicada por el parámetro shift) del gráfico correspondiente.

```
datetime  iTime(
   const string         symbol,          // símbolo
   ENUM_TIMEFRAMES      timeframe,       // periodo
   int                  shift            // desplazamiento
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

shift

[in] Índice del valor obtenido de la serie temporal (desplazamiento respecto a la barra actual en el número indicado de barras hacia atrás).

Valor devuelto

Valor de la hora y fecha de apertura de la barra (indicado con el parámetro shift) del gráfico correspondiente o 0 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Nota

La función siempre retorna los datos reales, para ello, envía en cada llamada una solicitud a las series temporales para el símbolo/periodo especificado. Esto significa que si no hay datos listos en la primera llamada de la función, puede requerir algún tiempo preparar el resultado de la ejecución.

La función no guarda los resultados de llamadas anteriores, no existe memoria caché local para el retorno rápido del valor

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- la hora y fecha cae en domingo
   datetime time=D'2018.06.10 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- no existe la barra de la hora y fecha indicadas, por eso iBarShift retornará el
   int bar_index=iBarShift(symbol,tf,time,exact);
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("Time of bar #%d is %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//PrintFormat(iTime(symbol,tf,bar_index));
//--- requerimos encontrar el índice de la barra para la hora y fecha especificadas, p
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
  }
//+------------------------------------------------------------------+
//| Retornamos el nombre para la semana                              |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(SUNDAY);
      break;
      case 1: day=EnumToString(MONDAY);
      break;
      case 2: day=EnumToString(TUESDAY);
      break;
      case 3: day=EnumToString(WEDNESDAY);
      break;
      case 4: day=EnumToString(THURSDAY);
      break;
      case 5: day=EnumToString(FRIDAY);
      break;
      default:day=EnumToString(SATURDAY);
      break;
     }
//---
   return day;
  }
/* Resultado:
  1 GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): bar index is 64 (exact=false)
   Time of bar #64 is 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):bar index is -1 (exact=true)
*/
```

Vea también

CopyTime, CopyRates

© 2000-2025, MetaQuotes Ltd.

iTickVolume

Retorna el valor del volumen de ticks de la barra (indicada por el parámetro shift) del gráfico correspondiente.

```
long  iTickVolume(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // periodo
   int                 shift            // desplazamiento
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

shift

[in] Índice del valor obtenido de la serie temporal (desplazamiento respecto a la barra actual en el número indicado de barras hacia atrás).

Valor devuelto

Valor del volumen de ticks de la barra (indicado con el parámetro shift) del gráfico correspondiente o 0 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Nota

La función siempre retorna los datos reales, para ello, envía en cada llamada una solicitud a las series temporales para el símbolo/periodo especificado. Esto significa que si no hay datos listos en la primera llamada de la función, puede requerir algún tiempo preparar el resultado de la ejecución.

La función no guarda los resultados de llamadas anteriores, no existe memoria caché local para el retorno rápido del valor.

Ejemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2025, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Vea también

CopyTickVolume, CopyRates

© 2000-2025, MetaQuotes Ltd.

iRealVolume

Retorna el valor del volumen real de la barra (indicada por el parámetro shift) del gráfico correspondiente.

```
long  iRealVolume(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // periodo
   int                 shift            // desplazamiento
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

shift

[in] Índice del valor obtenido de la serie temporal (desplazamiento respecto a la barra actual en el número indicado de barras hacia atrás).

Valor devuelto

Valor del volumen real de la barra (indicado con el parámetro shift) del gráfico correspondiente o 0 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Nota

La función siempre retorna los datos reales, para ello, envía en cada llamada una solicitud a las series temporales para el símbolo/periodo especificado. Esto significa que si no hay datos listos en la primera llamada de la función, puede requerir algún tiempo preparar el resultado de la ejecución.

La función no guarda los resultados de llamadas anteriores, no existe memoria caché local para el retorno rápido del valor.

Ejemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2025, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Vea también

CopyRealVolume, CopyRates

© 2000-2025, MetaQuotes Ltd.

iVolume

Retorna el valor del volumen de ticks de la barra (indicada por el parámetro shift) del gráfico correspondiente.

```
long  iVolume(
   const string         symbol,          // símbolo
   ENUM_TIMEFRAMES      timeframe,       // periodo
   int                  shift            // desplazamiento
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

shift

[in] Índice del valor obtenido de la serie temporal (desplazamiento respecto a la barra actual en el número indicado de barras hacia atrás).

Valor devuelto

Valor del volumen de ticks de la barra (indicado con el parámetro shift) del gráfico correspondiente o 0 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Nota

La función siempre retorna los datos reales, para ello, envía en cada llamada una solicitud a las series temporales para el símbolo/periodo especificado. Esto significa que si no hay datos listos en la primera llamada de la función, puede requerir algún tiempo preparar el resultado de la ejecución.

La función no guarda los resultados de llamadas anteriores, no existe memoria caché local para el retorno rápido del valor.

Ejemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2025, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Vea también

CopyTickVolume, CopyRates

© 2000-2025, MetaQuotes Ltd.

iSpread

Retorna el valor del spread de la barra (indicada por el parámetro shift) del gráfico correspondiente.

```
long  iSpread(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // periodo
   int                 shift            // desplazamiento
   );
```

Parámetros

symbol

[in] Nombre del símbolo del instrumento. NULL designa el símbolo actual.

timeframe

[in] Periodo. Puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 designa el periodo del gráfico actual.

shift

[in] Índice del valor obtenido de la serie temporal (desplazamiento respecto a la barra actual en el número indicado de barras hacia atrás).

Valor devuelto

Valor del spread para la barra (indicado con el parámetro shift) del gráfico correspondiente o 0 en caso de error. Para obtener información adicional sobre el error, es necesario llamar la función GetLastError().

Nota

La función siempre retorna los datos reales, para ello, envía en cada llamada una solicitud a las series temporales para el símbolo/periodo especificado. Esto significa que si no hay datos listos en la primera llamada de la función, puede requerir algún tiempo preparar el resultado de la ejecución.

La función no guarda los resultados de llamadas anteriores, no existe memoria caché local para el retorno rápido del valor.

Ejemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2025, MetaQuotes Ltd.

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "   ,DoubleToString(open,Digits()),"\n",
           "High: "   ,DoubleToString(high,Digits()),"\n",
           "Low: "    ,DoubleToString(low,Digits()),"\n",
           "Close: "  ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "   ,IntegerToString(bars),"\n"
           );
  }
```

Vea también

CopySpread, CopyRates

© 2000-2025, MetaQuotes Ltd.

Símbolos personalizados

Funciones para la creación y edición de las propiedades de los símbolos personalizados.

Al conectar el terminal a un servidor comercial concreto, el usuario tiene la posibilidad de trabajar con las series temporales de los instrumentos financieros que ofrece el bróker. Los instrumentos financieros disponibles se muestran en una lista de símbolos en la ventana de Market Watch, un grupo aparte de funciones permite obtener información sobre las propiedades del símbolo, las sesiones comerciales y las actualizaciones de la profundidad de mercado.

El grupo de funciones presentado en este apartado permite crear sus propios símbolos personalizados. Para ello, se pueden usar los símbolos existentes del servidor comercial, archivos de texto o fuentes de datos externas.

Función	Acción
CustomSymbolCreate	Crea un símbolo personalizado con el nombre indicado en el grupo indicado
CustomSymbolDelete	Elimina el símbolo personalizado con el nombre indicado
CustomSymbolSetInteger	Establece para el símbolo personalizado el valor de propiedad de tipo de número entero
CustomSymbolSetDouble	Establece para el símbolo personalizado el valor de propiedad de tipo real
CustomSymbolSetString	Establece para el símbolo personalizado el valor de propiedad de tipo string
CustomSymbolSetMarginRate	Establece para el símbolo personalizado los coeficientes del margen de carga dependiendo del tipo y la dirección de la orden
CustomSymbolSetSessionQuote	Establece la hora de comienzo y finalización de la sesión de cotizaciones para los símbolos y el día de la semana indicados
CustomSymbolSetSessionTrade	Establece la hora de comienzo y finalización de la sesión de comercial para los símbolos y el día de la semana indicados
CustomRatesDelete	Elimina todas las barras en el intervalo temporal indicado de la historia de precios del instrumento personalizado
CustomRatesReplace	Sustituye totalmente la historia de precios del instrumento personalizado en el intervalo temporal indicado, con los datos de la matriz del tipo MqlRates
CustomRatesUpdate	Añade a la historia del instrumento personalizado las barras ausentes y sustituye las existentes con datos de la matriz del tipo MqlRates
CustomTicksAdd	Añade a la historia de precios del instrumento personalizado los datos de la matriz del tipo MqlTick. El símbolo personalizado debe ser elegido en la ventana de MarketWatch (Observación del mercado)

© 2000-2025, MetaQuotes Ltd.

Función	Acción
CustomTicksDelete	Elimina todos los ticks en el intervalo temporal indicado de la historia de precios del instrumento personalizado
CustomTicksReplace	Sustituye totalmente la historia de precios del instrumento personalizado en el intervalo temporal indicado, con los datos de la matriz del tipo MqlTick
CustomBookAdd	Transmite el estado de la profundidad de mercado de un instrumento personalizado

© 2000-2025, MetaQuotes Ltd.

CustomSymbolCreate

Crea un símbolo personalizado con el nombre indicado en el grupo indicado.

```
bool  CustomSymbolCreate(
   const string      symbol_name,          // nombre del símbolo personalizado
   const string      symbol_path="",       // nombre del grupo en el que se creará el sí
   const string      symbol_origin=NULL    // nombre del símbolo sobre cuya base se crea
   );
```

Parámetros

symbol_name

[in] Nombre del símbolo personalizado. No debe contener el grupo o subgrupo en el que se encuentra el símbolo.

symbol_path=""

[in] Nombre del grupo en el que se crea el símbolo.

symbol_origin=NULL

[in] Nombre del símbolo del que se copiarán las propiedades del símbolo personalizado creado. Después de crear un símbolo personalizado, podremos cambiar cualquier propiedad por el valor necesario con las funciones correspondientes.

Valor retornado

true en el caso de éxito, de lo contrario, false. Para obtener información sobre el error, necesitamos llamar la función GetLastError().

Observación

Todos los símbolos personalizados se crean en el apartado especial Custom. Si no se ha establecido el nombre del grupo (el parámetro *symbol_path* en la función CustomSymbolCreate contiene una línea vacía o NULL), el símbolo personalizado se creará en la carpeta raíz del apartado Custom. Aquí podemos hacer una analogía con un sistema de archivos, donde todos los grupos-subgrupos pueden comprenderse como carpetas-subcarpetas

Tanto el nombre del símbolo como el del grupo se indicará solo con letras latinas, sin signos de puntuación, espacios en blanco o símbolos especiales (están permitidos ".", "_", "&" y "#"). No se recomienda usar en el nombre los símbolos <, >, :, ", /, |, ?, *.

El nombre del símbolo personalizado deberá ser único, independientemente del nombre del grupo en el que se cree. Si ya existe un símbolo con ese nombre, la función CustomSymbolCreate() retornará false, y la siguiente llamada de GetLastError() dará el código de error 5300 (ERR_NOT_CUSTOM_SYMBOL) o 5304 (ERR_CUSTOM_SYMBOL_EXIST).

La longitud del nombre del símbolo no deberá superar los 31 caracteres, en caso contrario, CustomSymbolCreate() retornará false y se generará el error 5302 - ERR_CUSTOM_SYMBOL_NAME_LONG.

El parámetro *symbol_path* se puede indicar de dos formas:

- solo el nombre del grupo sin el nombre del símbolo personalizado, por ejemplo, "CFD\\Metals". Es mejor usar precisamente esta opción, para evitar errores.

© 2000-2025, MetaQuotes Ltd.

- o bien el nombre <grupo> + separador de grupos "\\"+<nombre del símbolo personalizado>, por ejemplo, "CFD\\Metals\\Platinum". En este caso, el nombre del grupo deberá terminar con el nombre exacto del símbolo personalizado. En el caso de incompatibilidad, el símbolo personalizado se creará igualmente, pero no en el grupo en el que se pretendía. Por ejemplo, si *symbol_path*="CFD\\Metals**P**latinum" y *symbol_name*="**p**latinum" (error en el registro), se creará un símbolo personalizado con el nombre "platinum" en el grupo "Custom\CFD\Metals\Platinum". En este caso, además, la función SymbolInfoGetString("platinum",SYMBOL_PATH) retornará el valor "Custom\CFD\Metals\Platinum\platinum".

Es necesario tener en cuenta que la propiedad SYMBOL_PATH retorna la ruta junto con el nombre del símbolo al final. Por eso, no podemos copiarla simplemente sin cambios, si deseamos crear el símbolo personalizado precisamente en el mismo grupo. En este caso, será necesario cortar el nombre del símbolo, para así no obtener el resultado explicado anteriormente.

Si hemos establecido como parámetro *symbol_origin* un símbolo inexistente, el símbolo personalizado se creará vacío, como si el parámetro *symbol_origin* no hubiera sido indicado. En este caso, además, se generará el error 4301 - ERR_MARKET_UNKNOWN_SYMBOL.

La longitud del parámetro *symbol_path* no deberá superar los 127 caracteres, incluyendo "Custom\\", los separadores de grupos "\\" y el nombre del símbolo, si este se ha indicado al final.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                          CustomSymbolCreate.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // nombre del símbolo personalizado
#define   CUSTOM_SYMBOL_PATH     "Forex"        // nombre del grupo en el que se crea
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // nombre del símbolo a partir del cu

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- si no se ha podido crear el símbolo personalizado, informaremos de ello en el re
   if(!CustomSymbolCreate(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL_ORIGIN
     {
      Print("CustomSymbolCreate() failed. Error ", GetLastError());
      return;
     }

//--- comprobamos la existencia del símbolo creado y obtenemos el grupo en el que se h
```

© 2000-2025, MetaQuotes Ltd.

```
   bool   custom= false;
   bool   exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   string path  = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_PATH);

//--- imprimimos en el registro el resultado de la creación del símbolo y el nombre de
   PrintFormat("Custom symbol '%s' created\n"+
               "Symbol '%s' is exist: %s\n"+
               "Symbol '%s' is custom: %s\n"+
               "Path specified in the settings: '%s'\n"+
               "Path returned from the 'SYMBOL_PATH' property: '%s'",
               CUSTOM_SYMBOL_NAME,
               CUSTOM_SYMBOL_NAME, (string)exist,
               CUSTOM_SYMBOL_NAME, (string)custom,
               CUSTOM_SYMBOL_PATH,
               path);

//--- esperamos dos segundos y borramos el símbolo creado con un mensaje sobre el resu
   Sleep(2000);
   ResetLastError();
   bool deleted = CustomSymbolDelete(CUSTOM_SYMBOL_NAME);
   Print(deleted ? StringFormat("Custom symbol '%s' removed", CUSTOM_SYMBOL_NAME) : St
   /*
   resultado:
   Custom symbol 'EURUSD.C' created
   Symbol 'EURUSD.C' is exist: true
   Symbol 'EURUSD.C' is custom: true
   Path specified in the settings: 'Forex'
   Path returned from the 'SYMBOL_PATH' property: 'Custom\Forex\EURUSD.C'
   Custom symbol 'EURUSD.C' removed
   */
  }
```

Ver también

SymbolName, SymbolSelect, CustomSymbolDelete

© 2000-2025, MetaQuotes Ltd.

CustomSymbolDelete

Elimina el símbolo personalizado con el nombre indicado.

```
bool  CustomSymbolDelete(
   const string     symbol_name          // nombre del símbolo personalizado
   );
```

Parámetros

symbol

[in] Nombre del símbolo personalizado. No deberá coincidir con el nombre de un símbolo ya existente.

Valor devuelto

true - en caso de éxito, de lo contrario, false. Para obtener la información sobre el error, hay que llamar la función GetLastError().

Nota

Un símbolo representado en la Observación de Mercado (Market Watch) o del que haya un gráfico abierto, no podrá ser eliminado.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                          CustomSymbolDelete.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // nombre del símbolo personalizado
#define   CUSTOM_SYMBOL_PATH     "Forex"        // nombre del grupo en el que se crea
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // nombre del símbolo a partir del cu

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- si no se ha podido crear el símbolo personalizado, informaremos de ello en el re
   if(!CustomSymbolCreate(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL_ORIGIN
     {
      Print("CustomSymbolCreate() failed. Error ", GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- comprobamos la existencia del símbolo creado e imprimimos el resultado en el reg
   bool custom= false;
   bool exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   PrintFormat("Custom symbol '%s' exists: %s", CUSTOM_SYMBOL_NAME, (string)exist);

//--- esperamos dos segundos y borramos el símbolo creado con un mensaje sobre el resu
   Sleep(2000);
   ResetLastError();
   bool deleted = CustomSymbolDelete(CUSTOM_SYMBOL_NAME);
   Print(deleted ? StringFormat("Custom symbol '%s' removed", CUSTOM_SYMBOL_NAME) : St

//--- comprobamos la existencia del símbolo creado e imprimimos el resultado en el reg
   exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   PrintFormat("Custom symbol '%s' exists: %s", CUSTOM_SYMBOL_NAME, (string)exist);
   /*
   resultado:
   Custom symbol 'EURUSD.C' exists: true
   Custom symbol 'EURUSD.C' removed
   Custom symbol 'EURUSD.C' exists: false
   */
  }
```

Vea también

SymbolName, SymbolSelect, CustomSymbolCreate

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetInteger

Establece para el símbolo personalizado el valor de propiedad de tipo de número entero.

```
bool  CustomSymbolSetInteger(
   const string               symbol_name,      // nombre del símbolo
   ENUM_SYMBOL_INFO_INTEGER   property_id,      // identificador de la propiedad
   long                       property_value    // valor de la propiedad
   );
```

Parámetros

symbol_name

[in] Nombre del símbolo personalizado.

property_id

[in] Identificador de la propiedad del símbolo. El valor puede ser uno de los valores de la enumeración ENUM_SYMBOL_INFO_INTEGER.

property_value

[in] Variable del tipo long, que contiene el valor de la propiedad.

Valor devuelto

true - en caso de éxito, de lo contrario, false. Para obtener la información sobre el error, hay que llamar la función GetLastError().

Nota

La historia de minutos y ticks del símbolo personalizado se eliminará por completo, si en las especificaciones del símbolo se cambia cualquiera de estas propiedades:

- SYMBOL_CHART_MODE - tipo de precio utilizado para la construcción de las barras (Bid o Last)
- SYMBOL_DIGITS - número de dígitos decimales para la representación del precio

Después de eliminar la historia del símbolo personalizado, el terminal intentará crear una nueva historia usando las propiedades actualizadas. Lo mismo sucede al cambiar manualmente las propiedades del símbolo personalizado.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                      CustomSymbolSetInteger.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // nombre del símbolo personalizado
#define   CUSTOM_SYMBOL_PATH     "Forex"        // nombre del grupo en el que se crea
```

© 2000-2025, MetaQuotes Ltd.

```
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // nombre del símbolo a partir del cua

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- obtenemos e imprimimos en el registro las propiedades del símbolo en base al cua
//--- (modo de ejecución de las transacciones, nivel de ajuste de las órdenes Stop y c
   ENUM_SYMBOL_TRADE_EXECUTION origin_exe_mode = (ENUM_SYMBOL_TRADE_EXECUTION)SymbolIn
   int origin_stops_level = (int)SymbolInfoInteger(CUSTOM_SYMBOL_ORIGIN, SYMBOL_TRADE_
   int origin_freeze_level= (int)SymbolInfoInteger(CUSTOM_SYMBOL_ORIGIN, SYMBOL_TRADE_

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Deal execution mode: %s\n  Stops Level: %d\n  Freeze Level: %d",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               StringSubstr(EnumToString(origin_exe_mode), 23), origin_stops_level, or

//--- establecemos estas propiedades del símbolo personalizado en valores diferentes
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_EXEMODE, SYMBOL_TRADE
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_STOPS_LEVEL, 10);
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_FREEZE_LEVEL, 3);

//--- si se ha producido un error al configurar alguna de las propiedades, mostraremos
   if(!res)
      Print("CustomSymbolSetInteger() failed. Error ", GetLastError());

//--- obtenemos e imprimimos en el registro las propiedades modificadas del símbolo pe
//--- (modo de ejecución de las transacciones, nivel de ajuste de las órdenes Stop y c
   ENUM_SYMBOL_TRADE_EXECUTION custom_exe_mode = (ENUM_SYMBOL_TRADE_EXECUTION)SymbolIn
   int custom_stops_level = (int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_ST
   int custom_freeze_level= (int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_FR

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Deal execution mode: %s\n  Stops Level: %d\n  Freeze Level: %d",
               CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN,
               StringSubstr(EnumToString(custom_exe_mode), 23), custom_stops_level, cu

//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
```

© 2000-2025, MetaQuotes Ltd.

```
//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- pulsando Del, eliminaremos el símbolo personalizado creado
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Deal execution mode: INSTANT
     Stops Level: 0
     Freeze Level: 0
   Custom symbol 'EURUSD.C' based on 'EURUSD':
     Deal execution mode: MARKET
     Stops Level: 10
     Freeze Level: 3
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizado
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 530
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
```

Vea también

SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetDouble

Establece para el símbolo personalizado el valor de propiedad de tipo real.

```
bool  CustomSymbolSetDouble(
   const string              symbol_name,      // nombre del símbolo
   ENUM_SYMBOL_INFO_DOUBLE   property_id,      // identificador de la propiedad
   double                    property_value    // valor de la propiedad
   );
```

Parámetros

symbol_name

[in] Nombre del símbolo personalizado.

property_id

[in] Identificador de la propiedad del símbolo. El valor puede ser uno de los valores de la enumeración ENUM_SYMBOL_INFO_DOUBLE.

property_value

[in] Variable del tipo double, que contiene el valor de la propiedad.

Valor devuelto

true - en caso de éxito, de lo contrario, false. Para obtener la información sobre el error, hay que llamar la función GetLastError().

Nota

La historia de minutos y ticks del símbolo personalizado se eliminará por completo, si en las especificaciones del símbolo se cambia cualquiera de estas propiedades:

- SYMBOL_POINT - valor de un punto
- SYMBOL_TRADE_TICK_SIZE - valor de un tick, que especifica el cambio mínimo permitido del precio
- SYMBOL_TRADE_TICK_VALUE - coste del cambio de precio en un tick para una posición con beneficios

Después de eliminar la historia del símbolo personalizado, el terminal intentará crear una nueva historia usando las propiedades actualizadas. Lo mismo sucede al cambiar manualmente las propiedades del símbolo personalizado.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                      CustomSymbolSetDouble.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // nombre del símbolo personalizado
#define   CUSTOM_SYMBOL_PATH     "Forex"        // nombre del grupo en el que se crea
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // nombre del símbolo a partir del cu

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBO

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- obtenemos e imprimimos en el registro las propiedades del símbolo en base al cu
//--- (volumen mínimo, volumen máximo, paso de cambio de volumen mínimo para cerrar u
   double origin_vol_min = SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN, SYMBOL_VOLUME_MIN);
   double origin_vol_max = SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN, SYMBOL_VOLUME_MAX);
   double origin_vol_step= SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN, SYMBOL_VOLUME_STEP);

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Volume Min: %.2f\n  Volume Max: %.2f\n  Volume Step: %.2f",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               origin_vol_min, origin_vol_max, origin_vol_step);

//--- establecemos estas propiedades del símbolo personalizado en valores diferentes
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MIN, 0.1);
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MAX, 1000);
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_STEP, 0.1);

//--- si se ha producido un error al configurar alguna de las propiedades, mostraremo
   if(!res)
      Print("CustomSymbolSetDouble() failed. Error ", GetLastError());

//--- obtenemos e imprimimos en el registro las propiedades modificadas del símbolo pe
//--- (volumen mínimo, volumen máximo, paso de cambio de volumen mínimo para cerrar u
   double custom_vol_min = SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MIN);
   double custom_vol_max = SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MAX);
   double custom_vol_step= SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_STEP);

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Volume Min: %.2f\n  Volume Max: %.2f\n  Volume Step: %.2f",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               custom_vol_min, custom_vol_max, custom_vol_step);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex

//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- pulsando Del, eliminaremos el símbolo personalizado creado
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Volume Min: 0.01
     Volume Max: 500.00
     Volume Step: 0.01
   Custom symbol 'EURUSD' based on 'EURUSD.C':
     Volume Min: 0.10
     Volume Max: 1000.00
     Volume Step: 0.10
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizado
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 530
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
```

Vea también

SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetString

Establece para el símbolo personalizado el valor de propiedad de tipo string.

```
bool  CustomSymbolSetString(
   const string              symbol_name,      // nombre del símbolo
   ENUM_SYMBOL_INFO_STRING   property_id,      // identificador de la propiedad
   string                    property_value    // valor de la propiedad
   );
```

Parámetros

symbol_name

[in] Nombre del símbolo personalizado.

property_id

[in] Identificador de la propiedad del símbolo. El valor puede ser uno de los valores de la enumeración ENUM_SYMBOL_INFO_STRING.

property_value

[in] Variable del tipo string, que contiene el valor de la propiedad.

Valor devuelto

true - en caso de éxito, de lo contrario, false. Para obtener la información sobre el error, hay que llamar la función GetLastError().

Nota

La historia de minutos y ticks del símbolo personalizado se eliminará por completo, si en las especificaciones del símbolo se cambia la propiedad SYMBOL_FORMULA, que establece la fórmula para la construcción del precio del instrumento personalizado. Después de eliminar la historia del símbolo personalizado, el terminal intentará crear una nueva historia según la fórmula nueva. Lo mismo sucede al cambiar manualmente la fórmula del símbolo personalizado.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                        CustomSymbolSetString.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // nombre del símbolo personalizado
#define   CUSTOM_SYMBOL_PATH     "Forex"        // nombre del grupo en el que se crea
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // nombre del símbolo a partir del cu

//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- obtenemos e imprimimos en el registro las propiedades del símbolo en base al cua
//--- (volumen mínimo, volumen máximo, paso de cambio de volumen mínimo para cerrar un
   string origin_basis    = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_BASIS);
   string origin_category = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_CATEGORY);
   string origin_formula  = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_FORMULA);

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Basis: %s\n  Category: %s\n  Formula: %s",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               origin_basis, origin_category, origin_formula);

//--- establecemos estas propiedades del símbolo personalizado en valores diferentes
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_BASIS, CUSTOM_SYMBOL_ORIGIN);
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_CATEGORY, "FX");
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_FORMULA, ("1.0 / "+CUSTOM_SY

//--- si se ha producido un error al configurar alguna de las propiedades, mostraremos
   if(!res)
      Print("CustomSymbolSetString() failed. Error ", GetLastError());

//--- obtenemos e imprimimos en el registro las propiedades modificadas del símbolo pe
//--- (volumen mínimo, volumen máximo, paso de cambio de volumen mínimo para cerrar un
   string custom_basis    = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_BASIS);
   string custom_category = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_CATEGORY);
   string custom_formula  = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_FORMULA);

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Basis: %s\n  Category: %s\n  Formula: %s",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               custom_basis, custom_category, custom_formula);

//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex

//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Sleep(16);
       //--- pulsando Del, eliminaremos el símbolo personalizado creado
       if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
         {
          if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
             PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
          break;
         }
      }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Basis:
     Category:
     Formula:
   Custom symbol 'EURUSD' based on 'EURUSD.C':
     Basis: EURUSD
     Category: FX
     Formula: 1.0 / EURUSD
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizado
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 530
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
```

Vea también

SymbolInfoString

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetMarginRate

Establece para el símbolo personalizado las tasas del margen de carga dependiendo del tipo y la dirección de la orden.

```
bool  CustomSymbolSetMarginRate(
   const string       symbol_name,              // nombre del símbolo
   ENUM_ORDER_TYPE    order_type,               // tipo de orden
   double             initial_margin_rate,      // tasa de margen de carga inicial
   double             maintenance_margin_rate   // tasa de margen de carga de mantenir
   );
```

Parámetros

symbol_name

[in] Nombre del símbolo personalizado.

order_type

[in] Tipo de orden.

initial_margin_rate

[in] Variable del tipo double con el valor de la tasa de margen de carga inicial. El margen inicial es la cantidad mínima para ejecutar una transacción con un volumen de 1 lote en la dirección existente. Al multiplicar la tasa por el margen inicial, podemos obtener la cantidad de recursos que se reservará en la cuenta al colocar una orden del tipo indicado.

maintenance_margin_rate

[in] Variable del tipo double con el valor de la tasa de margen de carga de mantenimiento. El margen de mantenimiento es la cantidad mínima para mantener una posición abierta con un volumen de 1 lote en la dirección existente. Al multiplicar la tasa por el margen de mantenimiento, podemos obtener la cantidad de recursos que se reservará en la cuenta después de activarse una orden del tipo indicado.

Valor devuelto

true - en caso de éxito, de lo contrario, false. Para obtener la información sobre el error, hay que llamar la función GetLastError().

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                    CustomSymbolSetMarginRate.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // nombre del símbolo personalizado
```

© 2000-2025, MetaQuotes Ltd.

```
#define   CUSTOM_SYMBOL_PATH     "Forex"         // nombre del grupo en el que se crea
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()        // nombre del símbolo a partir del cu

#define   INITIAL_MARGIN_RATE       1.5          // coeficiente de cobro del margen in
#define   MAINTENANCE_MARGIN_RATE   1.5          // coeficiente de cobro del margen de

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBO

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- obtenemos e imprimimos en el registro las propiedades del símbolo en base al cu
//--- (coeficientes de cobro de margen inicial y de mantenimiento para órdenes Buy y S
   double initial_margin_rate_buy=0;
   double maintenance_margin_rate_buy=0;
   double initial_margin_rate_sell=0;
   double maintenance_margin_rate_sell=0;

   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_ORIGIN, ORDER_TYPE_BUY, initial_margin_rate_b
      return;
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_ORIGIN, ORDER_TYPE_SELL,initial_margin_rate_s
      return;

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Buy order initial margin rate: %f\n  Buy order maintenance margin ra
               "  Sell order initial margin rate: %f\n  Sell order maintenance margin
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               initial_margin_rate_buy, maintenance_margin_rate_buy, initial_margin_ra

//--- establecemos estas propiedades del símbolo personalizado en valores diferentes
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_BUY, INITIAL_MARGIN_
   res &=CustomSymbolSetMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_SELL,INITIAL_MARGIN_

//--- si se ha producido un error al configurar alguna de las propiedades, mostraremos
   if(!res)
      Print("CustomSymbolSetMarginRate() failed. Error ", GetLastError());

//--- obtenemos e imprimimos en el registro las propiedades modificadas del símbolo pe
//--- (coeficientes de cobro de margen inicial y de mantenimiento para órdenes Buy y S
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_BUY, initial_margin_rate_buy
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_SELL,initial_margin_rate_sel
      return;

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Buy order initial margin rate: %f\n  Buy order maintenance margin ra
               "  Sell order initial margin rate: %f\n  Sell order maintenance margin
               CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN,
               initial_margin_rate_buy, maintenance_margin_rate_buy, initial_margin_ra

//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex

//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- pulsando Del, eliminaremos el símbolo personalizado creado
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Buy order initial margin rate: 1.000000
     Buy order maintenance margin rate: 0.000000
     Sell order initial margin rate: 1.000000
     Sell order maintenance margin rate: 0.000000
   Custom symbol 'EURUSD.C' based on 'EURUSD':
     Buy order initial margin rate: 1.500000
     Buy order maintenance margin rate: 1.500000
     Sell order initial margin rate: 1.500000
     Sell order maintenance margin rate: 1.500000
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizado
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 530
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Retorna  los coeficientes de cobro de margen                     |
//+------------------------------------------------------------------+
bool GetSymbolMarginRate(const string symbol, const ENUM_ORDER_TYPE order_type, double
  {
   ResetLastError();
   if(!SymbolInfoMarginRate(symbol, order_type, initial_margin_rate, maintenance_margi
     {
      PrintFormat("%s: SymbolInfoMarginRate(%s, %s) failed. Error %d",__FUNCTION__, sy
      return false;
     }
   return true;
  }
```

© 2000-2025, MetaQuotes Ltd.

Vea también

SymbolInfoMarginRate

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetSessionQuote

Establece la hora de comienzo y finalización de la sesión de cotizaciones para los símbolos y el día de la semana indicados.

```
bool  CustomSymbolSetSessionQuote(
   const string      symbol_name,          // nombre del símbolo
   ENUM_DAY_OF_WEEK  day_of_week,          // día de la semana
   uint              session_index,        // número de la sesión
   datetime          from,                 // hora de comienzo de la sesión
   datetime          to                    // hora de finalización de la sesión
   );
```

Parámetros

symbol_name

[in] Nombre del símbolo personalizado.

ENUM_DAY_OF_WEEK

[in] Día de la semana, valor de la enumeración ENUM_DAY_OF_WEEK.

uint

[in] Número ordinal de la sesión para la que se necesita establecer la hora de comienzo y finalización. La indexación comienza a partir de 0.

from

[in] Hora del comienzo de la sesión en segundos, desde las 00 horas 00 minutos, el valor de la fecha en la variable será ignorado.

to

[in] Hora de la finalización de la sesión en segundos, desde las 00 horas 00 minutos, el valor de la fecha en la variable será ignorado.

Valor devuelto

true - en caso de éxito, de lo contrario, false. Para obtener la información sobre el error, hay que llamar la función GetLastError().

Nota

Si la sesión con el *session_index* indicado ya existe, la función simplemente editará el comienzo y la finalización de la sesión.

Si se han transmitido para la sesión parámetros cero de comienzo y finalización, es decir, se han establecido *from=0* y *to=0*, entonces la sesión existente con el índice *session_index* se eliminará, y la propia numeración de la sesión se desplazará hacia abajo.

Solo se pueden añadir sesiones de forma consecutiva, es decir, una sesión con el índice *session_index=1* se puede añadir solo en el caso de que exista una sesión con un índice igual a 0. Si se rompe esta regla, la nueva sesión no se creará, y la propia función retornará el valor false.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                  CustomSymbolSetSessionQuote.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"           // nombre del símbolo persona
#define   CUSTOM_SYMBOL_PATH     "Forex"                 // nombre del grupo en el que
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()                // nombre del símbolo a parti

#define   SESSION_0_FROM         D'1970.01.01 00:15:00'  // hora de comienzo de la ses
#define   SESSION_0_TO           D'1970.01.01 11:59:00'  // hora de finalización de la
#define   SESSION_1_FROM         D'1970.01.01 12:15:00'  // hora de comienzo de la ses
#define   SESSION_1_TO           D'1970.01.01 23:59:00'  // hora de finalización de la

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- imprimimos el encabezado con el símbolo básico y el índice de sesión y
//--- en un ciclo por los días de la semana de lunes a viernes, imprimimos en el regis
   for(int session=0; session<2; session++)
     {
      PrintFormat("Quote session %d of '%s' symbol from which the custom '%s' was crea
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_ORIGIN, (ENUM_DAY_OF_WEEK)day_of_we
     }

//--- en un ciclo por las dos sesiones
   bool res=true;
   for(int session=0; session<2; session++)
     {
      datetime from = SESSION_0_FROM;
      datetime to   = SESSION_0_TO;
      if(session>0)
        {
         from = SESSION_1_FROM;
         to   = SESSION_1_TO;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      //--- para el símbolo personalizado para cada día de la semana establecemos la h
      ResetLastError();
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         res &=CustomSymbolSetSessionQuote(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_
     }

//--- si se ha producido un error al configurar alguna de las sesiones, mostraremos un
   if(!res)
      Print("CustomSymbolSetSessionQuote() failed. Error ", GetLastError());

//--- imprimimos el encabezado con el símbolo personalizado y el índice de sesión y
//--- en un ciclo por los días de la semana de lunes a viernes, imprimimos en el regis
   for(int session=0; session<2; session++)
     {
      PrintFormat("Quote session %d of custom symbol '%s' based on '%s'", session, CUS
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_of_week
     }

//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- pulsando Del, eliminaremos el símbolo personalizado creado
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   Quote session 0 of 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     00:15 - 23:55
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
   Quote session 1 of 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     Session not set
   - Tuesday    Session not set
   - Wednesday  Session not set
   - Thursday   Session not set
```

© 2000-2025, MetaQuotes Ltd.

```
   - Friday     Session not set
   Quote session 0 of custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     00:15 - 11:59
   - Tuesday    00:15 - 11:59
   - Wednesday  00:15 - 11:59
   - Thursday   00:15 - 11:59
   - Friday     00:15 - 11:59
   Quote session 1 of custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     12:15 - 23:59
   - Tuesday    12:15 - 23:59
   - Wednesday  12:15 - 23:59
   - Thursday   12:15 - 23:59
   - Friday     12:15 - 23:59
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizado
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 530
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mostramos en el registro la hora de inicio y finalización de     |
//| la sesión de cotización para los símbolos y días indicados       |
//+------------------------------------------------------------------+
void SymbolInfoSessionQuotePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- declaramos las variables para registrar el inicio y el final de la sesión de cot
   datetime date_from;  // hora de inicio de la sesión
   datetime date_to;    // hora de finalización de la sesión

//--- creamos el nombre del día de la semana a partir de una constante de enumeración
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- obtenemos los datos de la sesión de cotización
   if(!SymbolInfoSessionQuote(symbol, day_of_week, session_index, date_from, date_to))
     {
      int err=GetLastError();
      string message=(err==4307 ? StringFormat("- %-10s Session not set", week_day) :
                      StringFormat("SymbolInfoSessionQuote(%s, %s, session %d) failed.
      Print(message);
      return;
     }

//--- mostramos en el registro los datos de la sesión de cotización
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

Vea también

SymbolInfoSessionQuote, Información sobre el instrumento, TimeToStruct, Estructura de la fecha

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetSessionTrade

Establece la hora de comienzo y finalización de la sesión comercial para los símbolos y el día de la semana indicados.

```
bool  CustomSymbolSetSessionTrade(
   const string      symbol_name,          // nombre del símbolo
   ENUM_DAY_OF_WEEK  day_of_week,          // día de la semana
   uint              session_index,        // número de la sesión
   datetime          from,                 // hora de comienzo de la sesión
   datetime          to                    // hora de finalización de la sesión
   );
```

Parámetros

symbol_name

[in] Nombre del símbolo personalizado.

ENUM_DAY_OF_WEEK

[in] Día de la semana, valor de la enumeración ENUM_DAY_OF_WEEK.

uint

[in] Número ordinal de la sesión para la que se necesita establecer la hora de comienzo y finalización. La indexación comienza a partir de 0.

from

[in] Hora del comienzo de la sesión en segundos, desde las 00 horas 00 minutos, el valor de la fecha en la variable será ignorado.

to

[in] Hora de la finalización de la sesión en segundos, desde las 00 horas 00 minutos, el valor de la fecha en la variable será ignorado.

Valor devuelto

true - en caso de éxito, de lo contrario, false. Para obtener la información sobre el error, hay que llamar la función GetLastError().

Nota

Si la sesión con el *session_index* indicado ya existe, la función simplemente editará el comienzo y la finalización de la sesión.

Si se han transmitido para la sesión parámetros cero de comienzo y finalización, es decir, se han establecido *from=0* y *to=0*, entonces la sesión existente con el índice *session_index* se eliminará, y la propia numeración de la sesión se desplazará hacia abajo.

Solo se pueden añadir sesiones de forma consecutiva, es decir, una sesión con el índice *session_index=1* se puede añadir solo en el caso de que exista una sesión con un índice igual a 0. Si se rompe esta regla, la nueva sesión no se creará, y la propia función retornará el valor false.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                  CustomSymbolSetSessionTrade.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"           // nombre del símbolo persona
#define   CUSTOM_SYMBOL_PATH     "Forex"                 // nombre del grupo en el que
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()                // nombre del símbolo a parti

#define   SESSION_0_FROM         D'1970.01.01 00:15:00'  // hora de comienzo de la ses
#define   SESSION_0_TO           D'1970.01.01 11:59:00'  // hora de finalización de la
#define   SESSION_1_FROM         D'1970.01.01 12:15:00'  // hora de comienzo de la ses
#define   SESSION_1_TO           D'1970.01.01 23:59:00'  // hora de finalización de la

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- imprimimos el encabezado con el símbolo básico y el índice de sesión y
//--- en un ciclo por los días de la semana de lunes a viernes, imprimimos en el regis
   for(int session=0; session<2; session++)
     {
      PrintFormat("Trade session %d of the '%s' symbol from which the custom '%s' was
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_ORIGIN, (ENUM_DAY_OF_WEEK)day_of_we
     }

//--- en un ciclo por las dos sesiones
   bool res=true;
   for(int session=0; session<2; session++)
     {
      datetime from = SESSION_0_FROM;
      datetime to   = SESSION_0_TO;
      if(session>0)
        {
         from = SESSION_1_FROM;
         to   = SESSION_1_TO;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      //--- para el símbolo personalizado para cada día de la semana establecemos la h
      ResetLastError();
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         res &=CustomSymbolSetSessionQuote(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_o
     }

//--- si se ha producido un error al configurar alguna de las sesiones, mostraremos un
   if(!res)
      Print("CustomSymbolSetSessionTrade() failed. Error ", GetLastError());

//--- imprimimos el encabezado con el símbolo personalizado y el índice de sesión y
//--- en un ciclo por los días de la semana de lunes a viernes, imprimimos en el regis
   for(int session=0; session<2; session++)
     {
      PrintFormat("Trade session %d of a custom symbol '%s' based on '%s'", session, C
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_of_week
     }

//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- pulsando Del, eliminaremos el símbolo personalizado creado
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   Trade session 0 of the 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     00:15 - 23:55
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
   Trade session 1 of the 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     Session not set
   - Tuesday    Session not set
   - Wednesday  Session not set
   - Thursday   Session not set
```

© 2000-2025, MetaQuotes Ltd.

```
   - Friday     Session not set
   Trade session 0 of a custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     00:15 - 11:59
   - Tuesday    00:15 - 11:59
   - Wednesday  00:15 - 11:59
   - Thursday   00:15 - 11:59
   - Friday     00:15 - 11:59
   Trade session 1 of a custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     12:15 - 23:59
   - Tuesday    12:15 - 23:59
   - Wednesday  12:15 - 23:59
   - Thursday   12:15 - 23:59
   - Friday     12:15 - 23:59
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizado
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 530
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Mostramos en el registro la hora de inicio y finalización de     |
//| sesión comercial para el símbolo y el día indicado               |
//+------------------------------------------------------------------+
void SymbolInfoSessionTradePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- declaramos las variables para registrar el inicio y el final de la sesión de cot
   datetime date_from;  // hora de inicio de la sesión
   datetime date_to;    // hora de finalización de la sesión

//--- creamos el nombre del día de la semana a partir de una constante de enumeración
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- obtenemos los datos de la sesión de cotización
   if(!SymbolInfoSessionTrade(symbol, day_of_week, session_index, date_from, date_to))
     {
      int err=GetLastError();
      string message=(err==4307 ? StringFormat("- %-10s Session not set", week_day) :
                      StringFormat("SymbolInfoSessionTrade(%s, %s, session %d) failed.
      Print(message);
      return;
     }

//--- mostramos en el registro los datos de la sesión comercial indicada
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

Vea también

SymbolInfoSessionTrade, Información sobre el instrumento, TimeToStruct, Estructura de la fecha

© 2000-2025, MetaQuotes Ltd.

CustomRatesDelete

Elimina todas las barras en el intervalo temporal indicado de la historia de precio del instrumento personalizado.

```
int  CustomRatesDelete(
   const string     symbol,        // nombre del símbolo
   datetime         from,          // desde qué fecha
   datetime         to             // hasta qué fecha
   );
```

Parámetros

symbol

[in] Nombre del instrumento personalizado.

from

[in] Hora de la primera barra en la historia de precio del diapasón indicado, que debe ser eliminada.

to

[in] Hora de la última barra en la historia de precio del diapasón indicado, que debe ser eliminada.

Valor devuelto

Número de barras eliminadas o bien -1 en caso de error.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                            CustomRatesDelete.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // nombre del símbolo personalizad
#define   CUSTOM_SYMBOL_PATH     "Forex"           // nombre del grupo en el que se cr
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // nombre del símbolo a partir del

#define   DATARATES_COUNT        4                 // número de barras a mostrar en el

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- obtenemos el número de barras del símbolo estándar
   int bars=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);

//--- obtenemos la matriz MqlRates los datos de todas las barras del marco temporal de
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_OR
      return;
     }

//--- establecemos los datos copiados en la historia de minutos del símbolo personaliz
   ResetLastError();
   if(CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates)<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- después de actualizar los datos históricos obtenemos el número de barras del sím
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- obtenemos en la matriz MqlRates los datos de todas las barras del marco temporal
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- imprimimos las últimas barras DATARATES_COUNT de la historia de minutos del símb
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:", DATARATES_COUNT)
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- borramos las dos penúltimas barras de datos de la historia de minutos del símbol
   datetime time_from= rates[bars-3].time;
   datetime time_to  = rates[bars-2].time;
   ResetLastError();
   int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, time_from, time_to);
   if(deleted<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("CustomRatesDelete(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
       return;
      }

//--- tras eliminar dos barras de datos históricos, obtendremos de nuevo el número de
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- obtenemos nuevamente los datos de todas las barras restantes del marco temporal
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- imprimimos las últimas barras DATARATES_COUNT de la historia de minutos actualiz
   PrintFormat("\nLast %d bars after applying CustomRatesDelete() with %d deleted bars
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- al presionar Supr, eliminamos el símbolo personalizado creado y sus datos
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- eliminamos los datos de las barras
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully d

         //--- eliminamos los datos de ticks
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- eliminamos el símbolo
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
```

© 2000-2025, MetaQuotes Ltd.

```
   Last 4 bars of the custom symbol's minute history:
                  [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 20:53:00 1.07341 1.07347 1.07336 1.07343            38        0
   [1] 2024.06.18 20:54:00 1.07344 1.07354 1.07344 1.07353            21        0
   [2] 2024.06.18 20:55:00 1.07353 1.07362 1.07351 1.07356            32        0
   [3] 2024.06.18 20:56:00 1.07356 1.07358 1.07352 1.07354            24        0

   Last 4 bars after applying CustomRatesDelete() with 2 deleted bars:
                  [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 20:51:00 1.07357 1.07358 1.07347 1.07349            25        0
   [1] 2024.06.18 20:52:00 1.07349 1.07350 1.07336 1.07341            31        0
   [2] 2024.06.18 20:53:00 1.07341 1.07347 1.07336 1.07343            38        0
   [3] 2024.06.18 20:56:00 1.07356 1.07358 1.07352 1.07354            24        0
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizado
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 530
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
```

Vea también

CustomRatesReplace, CustomRatesUpdate, CopyRates

© 2000-2025, MetaQuotes Ltd.

CustomRatesReplace

Sustituye totalmente la historia de precios del instrumento personalizado en el intervalo temporal indicado, con los datos de la matriz MqlRates.

```
int  CustomRatesReplace(
   const string     symbol,              // nombre del símbolo
   datetime         from,                // desde qué fecha
   datetime         to,                  // hasta qué fecha
   const MqlRates&  rates[],             // matriz con los datos que necesitamos aplica
   uint             count=WHOLE_ARRAY    // número de elementos que se usarán de la mat
   );
```

Parámetros

symbol

[in] Nombre del instrumento personalizado.

from

[in] Hora de la primera barra en la historia de precios del diapasón indicado, que debe ser actualizada.

to

[in] Hora de la última barra en la historia de precios del diapasón indicado, que debe ser actualizada.

rates[]

[in] Matriz de datos históricos del tipo MqlRates para el marco temporal M1.

count=WHOLE_ARRAY

[in] Número de elementos de la matriz *rates[]* que se usarán para la sustitución. El valor WHOLE_ARRAY indica que para la sustitución se debe usar todos los elementos de la matriz *rates[]*.

Valor devuelto

Número de barras actualizadas o bien -1 en caso de error.

Nota

Si una barra de la matriz *rates[]* se sale de los límites del diapasón, entonces será ignorada. Si ya disponemos de esa barra en la historia de precios y entra en el diapasón indicado, entonces será sustituida. El resto de las barras en la historia de precios actual fuera de los límites del diapasón indicado permanecerá sin cambios. Los datos en la matriz *rates[]* deberán ser correctos en cuanto a los precios OHLC, y la hora de apertura de las barras deberá corresponderse con el marco temporal M1.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                          CustomRatesReplace.mq5 |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // nombre del símbolo personalizado
#define   CUSTOM_SYMBOL_PATH     "Forex"           // nombre del grupo en el que se cr
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // nombre del símbolo a partir del

#define   DATARATES_COUNT        4                 // número de barras a mostrar en el

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- obtenemos el número de barras del símbolo estándar
   int bars=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);

//--- obtenemos la matriz MqlRates los datos de todas las barras del marco temporal de
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_OR
      return;
     }

//--- establecemos los datos copiados en la historia de minutos del símbolo personaliz
   ResetLastError();
   if(CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates)<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- después de actualizar los datos históricos obtenemos el número de barras del sím
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- obtenemos en la matriz MqlRates los datos de todas las barras del marco temporal
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- imprimimos las últimas barras DATARATES_COUNT de la historia de minutos del símb
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:", DATARATES_COUNT)
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- cambiamos las dos penúltimas barras de datos de la historia de minutos del símbo
   datetime time_from= rates[bars-3].time;
   datetime time_to  = rates[bars-2].time;

//--- en la matriz rates hacemos que todos los precios de las dos barras precedentes s
   rates[bars-3].high=rates[bars-3].open;
   rates[bars-3].low=rates[bars-3].open;
   rates[bars-3].close=rates[bars-3].open;

   rates[bars-2].high=rates[bars-2].open;
   rates[bars-2].low=rates[bars-2].open;
   rates[bars-2].close=rates[bars-2].open;

//--- sustituimos las barras existentes por los datos de la matriz modificada rates
   ResetLastError();
   int replaced=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(replaced<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- después de cambiar dos barras de datos históricos, obtendremos de nuevo el núme
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- obtenemos nuevamente los datos de todas las barras del marco temporal de minutos
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- imprimimos las últimas barras DATARATES_COUNT de la historia de minutos actualiz
   PrintFormat("\nLast %d bars after applying CustomRatesUpdate() with %d replaced bar
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- al presionar Supr, eliminamos el símbolo personalizado creado y sus datos
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- eliminamos los datos de las barras
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully d

         //--- eliminamos los datos de ticks
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- eliminamos el símbolo
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   Last 4 bars of the custom symbol's minute history:
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.07.29 13:37:00 1.08394 1.08396 1.08388 1.08390            16        1
   [1] 2024.07.29 13:38:00 1.08389 1.08400 1.08389 1.08398            35        1
   [2] 2024.07.29 13:39:00 1.08398 1.08410 1.08394 1.08410            29        1
   [3] 2024.07.29 13:40:00 1.08409 1.08414 1.08408 1.08414            14        1

   Last 4 bars after applying CustomRatesUpdate() with 250820 replaced bars:
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.07.29 13:37:00 1.08394 1.08396 1.08388 1.08390            16        1
   [1] 2024.07.29 13:38:00 1.08389 1.08389 1.08389 1.08389            35        1
   [2] 2024.07.29 13:39:00 1.08398 1.08398 1.08398 1.08398            29        1
   [3] 2024.07.29 13:40:00 1.08409 1.08414 1.08408 1.08414            14        1
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizado
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 530
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
```

Vea también

CustomRatesDelete, CustomRatesUpdate, CopyRates

© 2000-2025, MetaQuotes Ltd.

CustomRatesUpdate

Añade a la historia del instrumento personalizado las barras ausentes y sustituye las existentes con datos de la matriz del tipo MqlRates.

```
int  CustomRatesUpdate(
   const string     symbol,             // nombre del símbolo personalizado
   const MqlRates&  rates[],            // matriz con los datos que necesitamos aplica
   uint             count=WHOLE_ARRAY   // número de elementos que se usarán de la mat
   );
```

Parámetros

symbol

[in] Nombre del instrumento personalizado.

rates[]

[in] Matriz de datos históricos del tipo MqlRates para el marco temporal M1.

count=WHOLE_ARRAY

[in] Número de elementos de la matriz *rates[]* que se usarán para la actualización. El valor WHOLE_ARRAY indica que para la actualización se debe usar todos los elementos de la matriz *rates[]*.

Valor devuelto

Número de barras actualizadas o bien -1 en caso de error.

Nota

Si la barra de la matriz *rates[]* está ausente en la historia actual del instrumento persoanalizado, entonces se añade. Si ya existe tal barra, entonces se sustituye. El resto de la barras en la historia de precios actual permanecerán sin cambios. Los datos en la matriz *rates[]* deberán ser correctos en cuanto a los precios OHLC, y la hora de apertura de las barras deberá corresponderse con el marco temporal M1.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                            CustomRatesUpdate.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // nombre del símbolo personalizado
#define   CUSTOM_SYMBOL_PATH     "Forex"           // nombre del grupo en el que se cr
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // nombre del símbolo a partir del
```

© 2000-2025, MetaQuotes Ltd.

```
#define   DATARATES_COUNT        4                 // número de barras a mostrar en el

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- obtenemos e imprimimos en el registro el número de barras del símbolo estándar
   int bars_origin=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);
   PrintFormat("The symbol '%s' from which the custom '%s' was created has %d bars of

//--- obtenemos e imprimimos en el registro el número de barras del símbolo personaliz
   int bars_custom=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);
   PrintFormat("Custom symbol '%s' created from symbol '%s' has %d bars of minute hist

//--- obtenemos la matriz MqlRates los datos de todas las barras del marco temporal de
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars_origin, rates)!=bars_origin)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_OR
      return;
     }

//--- establecemos los datos copiados en la historia de minutos del símbolo personaliz
   ResetLastError();
   int updated=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(updated<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- obtenemos e imprimimos en el registro el número de barras del símbolo personaliz
   bars_custom=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);
   PrintFormat("\nAfter CustomRatesUpdate(), the custom symbol '%s' has %d bars of min

//--- obtenemos en la matriz MqlRates los datos de todas las barras del marco temporal
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars_custom, rates)!=bars_custom)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

//--- imprimimos las cuatro últimas barras de la historia de minutos del símbolo perso
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:", DATARATES_COUNT)
   ArrayPrint(rates, digits, NULL, bars_custom-DATARATES_COUNT, DATARATES_COUNT);

//--- cambiamos los datos en la matriz MqlRates por los calculados según la fórmula 1.
   for(int i=0; i<bars_custom; i++)
     {
      rates[i].open  =(rates[i].open !=0  ? 1.0 / rates[i].open  : rates[i].open);
      rates[i].high  =(rates[i].high !=0  ? 1.0 / rates[i].high  : rates[i].high);
      rates[i].low   =(rates[i].low  !=0  ? 1.0 / rates[i].low   : rates[i].low);
      rates[i].close =(rates[i].close!=0  ? 1.0 / rates[i].close : rates[i].close);
     }

//--- establecemos los datos modificados en la historia de minutos del símbolo persona
   ResetLastError();
   updated=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(updated<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- obtenemos de nuevo los datos de todas las barras del marco temporal de minutos d
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars_custom, rates)!=bars_custom)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- imprimimos las cuatro últimas barras de la historia de minutos actualizada del s
   Print("\nLast %d bars after changing the custom symbol calculation formula:", DATAR
   ArrayPrint(rates, digits, NULL, bars_custom-DATARATES_COUNT, DATARATES_COUNT);

//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- al presionar Supr, eliminamos el símbolo personalizado creado y sus datos
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- eliminamos los datos de las barras
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
```

© 2000-2025, MetaQuotes Ltd.

```
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully c

         //--- eliminamos los datos de ticks
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- eliminamos el símbolo
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   The symbol 'EURUSD' from which the custom 'EURUSD.C' was created has 250488 bars of
   Custom symbol 'EURUSD.C' created from symbol 'EURUSD' has 0 bars of minute history

   After CustomRatesUpdate(), the custom symbol 'EURUSD.C' has 250488 bars of minute h
   Last 4 bars of the custom symbol's minute history:
                  [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 11:14:00 1.07235 1.07239 1.07232 1.07239            24        0
   [1] 2024.06.18 11:15:00 1.07238 1.07239 1.07232 1.07235            44        0
   [2] 2024.06.18 11:16:00 1.07234 1.07238 1.07227 1.07234            37        0
   [3] 2024.06.18 11:17:00 1.07234 1.07234 1.07217 1.07225            41        0

   Last 4 bars after changing the custom symbol calculation formula:
                  [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 11:14:00 0.93253 0.93250 0.93256 0.93250            24        0
   [1] 2024.06.18 11:15:00 0.93251 0.93250 0.93256 0.93253            44        0
   [2] 2024.06.18 11:16:00 0.93254 0.93251 0.93260 0.93254            37        0
   [3] 2024.06.18 11:17:00 0.93254 0.93254 0.93269 0.93262            41        0
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizado
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 53(
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       error=GetLastError();
       if(error!=5304)
          PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
      }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
```

Vea también

CustomRatesReplace, CustomRatesDelete, CopyRates

© 2000-2025, MetaQuotes Ltd.

CustomTicksAdd

Añade a la historia de precios del instrumento personalizado los datos de la matriz del tipo MqlTick. El símbolo personalizado debe ser elegido en la ventana de MarketWatch (Observación del mercado).

```
int  CustomTicksAdd(
   const string     symbol,              // nombre del símbolo
   const MqlTick&   ticks[],             // matriz con los datos de ticks que necesitar
   uint             count=WHOLE_ARRAY    // número de elementos que se usarán de la mat
   );
```

Parámetros

symbol

[in] Nombre del instrumento personalizado.

ticks[]

[in] Matriz con los datos de ticks del tipo MqlTick, organizados en orden temporal ascendente, es decir, es necesario que ticks[k].time_msc <= ticks[n].time_msc, si k<n.

count=WHOLE_ARRAY

[in] Número de elementos de la matriz *ticks[]* que se usarán para la adición. El valor WHOLE_ARRAY indica que se debe usar todos los elementos de la matriz *ticks[]*.

Valor devuelto

Número de ticks añadidos o bien -1 en caso de error.

Nota

La función CustomTicksAdd solo funciona para los símbolos personalizados abiertos en la ventana MarketWatch (Observación del mercado). Si el símbolo no se ha elegido en MarketWatch, para insertar ticks será necesario usar CustomTicksReplace.

La función CustomTicksAdd permite transmitir los ticks como si llegaran desde el servidor del bróker. La información no se guarda directamente en el banco de datos de ticks, sino que es enviada a la ventana "Observación de mercado". Y ya desde ella, el terminal almacena los ticks en su base. Cuando la cantidad de información transferida por la llamada tiene un gran volumen, la función cambia su comportamiento para ahorrar recursos. Si se transfieren más de 256 ticks, los datos se dividen en dos partes. La primera de ellas (la grande) se carga de inmediato en la base de ticks (como hace CustomTicksReplace). La segunda, que consta de los últimos 128 ticks, se transfiere a la ventana "Observación de mercado" y, después de ello, es guardada por el terminal en la base de ticks.

La estructura MqlTick tiene dos campos con el valor de hora - time (hora del tick en segundos) y time_msc (hora del tick en milisegundos) que llevan la cuenta desde el 01 de enero del año 1970. El procesamiento de estos campos en los ticks añadidos se realiza según las normas siguientes en el orden indicado:

1. si el valor ticks[k].time_msc!=0, lo usaremos para rellenar el campo ticks[k].time, es decir, para el tick se muestra la hora ticks[k].time=ticks[k].time_msc/1000 (división en números enteros)
2. si ticks[k].time_msc==0 y ticks[k].time!=0, la hora en milisigundos resultará multiplicada por 1000, es decir, ticks[k].time_msc=ticks[k].time*1000

© 2000-2025, MetaQuotes Ltd.

3. si ticks[k].time_msc==0 y ticks[k].time==0, en estos campos se registra la hora actual del servidor comercial con una precisión de milisegundos en el momento de la llamada de la función CustomTicksAdd.

Si el valor de los campos ticks[k].bid, ticks[k].ask, ticks[k].last o ticks[k].volume es superior a cero, en el campo ticks[k].flags se escribirá una combinación de las banderas correspondientes:

- TICK_FLAG_BID - el tick ha cambiado el precio bid
- TICK_FLAG_ASK - el tick ha cambiado el precio ask
- TICK_FLAG_LAST - el tick ha cambiado el precio de la última transacción
- TICK_FLAG_VOLUME - el tick ha cambiado el volumen

Si el valor de algún campo es menor o igual a cero, la bandera que le corresponda no se anotará en el campo ticks[k].flags.

Las banderas TICK_FLAG_BUY y TICK_FLAG_SELL no se añaden a la historia del instrumento personalizado.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                CustomTicksAdd.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // nombre del símbolo personalizado
#define   CUSTOM_SYMBOL_PATH     "Forex"           // nombre del grupo en el que se cr
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // nombre del símbolo a partir del

#define   DATATICKS_TO_COPY      UINT_MAX          // cantidad de ticks copiados
#define   DATATICKS_TO_PRINT     20                // cantidad de ticks a mostrar en e

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- obtenemos en la matriz MqlTick los datos de los ticks del símbolo estándar
   MqlTick array[]={};
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;

//--- imprimimos la hora del primer y el último tick obtenidos del símbolo estándar
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- imprimimos en el registro DATATATICKS_TO_PRINT los últimos ticks del símbolo est
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':", DATATICKS_TO_PRINT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- añadimos el símbolo personalizado a la ventana MarketWatch (observación de merca
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- añadimos a la historia de precios del símbolo personalizado los datos de la mat
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'", ar
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms", add

//--- obtenemos en la matriz MqlTick los datos de los ticks del símbolo personalizado
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- imprimimos la hora del primer y el último tick obtenidos del símbolo personaliza
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- imprimimos en el registro DATATATICKS_TO_PRINT los últimos ticks del símbolo per
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':", DATATICKS_TO_PRINT,
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
```

© 2000-2025, MetaQuotes Ltd.

```
         continue;
       PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
      }

//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- al presionar Supr, eliminamos el símbolo personalizado creado y sus datos
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- eliminamos los datos de las barras
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully d

         //--- eliminamos los datos de ticks
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- eliminamos el símbolo
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   Requested 4294967295 ticks to download tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351183943 tic
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.20 21:18:12.010

   The last 20 ticks for the standard symbol 'EURUSD':
     351183923th Tick: 2024.06.20 21:17:46.380 Bid=1.07124 (Info tick)
     351183924th Tick: 2024.06.20 21:17:47.779 Ask=1.07125 Bid=1.07125 (Info tick)
     351183925th Tick: 2024.06.20 21:17:48.584 Ask=1.07124 Bid=1.07124 (Info tick)
     351183926th Tick: 2024.06.20 21:17:49.481 Ask=1.07125 (Info tick)
     351183927th Tick: 2024.06.20 21:17:49.985 Ask=1.07122 Bid=1.07122 (Info tick)
     351183928th Tick: 2024.06.20 21:17:50.482 Ask=1.07124 Bid=1.07124 (Info tick)
     351183929th Tick: 2024.06.20 21:17:51.584 Ask=1.07123 Bid=1.07123 (Info tick)
     351183930th Tick: 2024.06.20 21:17:52.786 Ask=1.07124 Bid=1.07124 (Info tick)
     351183931th Tick: 2024.06.20 21:17:53.487 Ask=1.07125 Bid=1.07125 (Info tick)
     351183932th Tick: 2024.06.20 21:17:53.989 Ask=1.07126 Bid=1.07126 (Info tick)
     351183933th Tick: 2024.06.20 21:17:55.789 Ask=1.07125 Bid=1.07125 (Info tick)
```

© 2000-2025, MetaQuotes Ltd.

```
     351183934th Tick: 2024.06.20 21:17:58.495 Ask=1.07126 Bid=1.07126 (Info tick)
     351183935th Tick: 2024.06.20 21:18:00.102 Bid=1.07126 (Info tick)
     351183936th Tick: 2024.06.20 21:18:00.698 Ask=1.07129 Bid=1.07129 (Info tick)
     351183937th Tick: 2024.06.20 21:18:03.699 Bid=1.07129 (Info tick)
     351183938th Tick: 2024.06.20 21:18:04.699 Ask=1.07128 Bid=1.07128 (Info tick)
     351183939th Tick: 2024.06.20 21:18:05.901 Ask=1.07129 Bid=1.07129 (Info tick)
     351183940th Tick: 2024.06.20 21:18:07.606 Ask=1.07128 Bid=1.07128 (Info tick)
     351183941th Tick: 2024.06.20 21:18:11.512 Ask=1.07127 Bid=1.07127 (Info tick)
     351183942th Tick: 2024.06.20 21:18:12.010 Ask=1.07126 Bid=1.07126 (Info tick)
   ...
   Start of adding 351183943 ticks to the history of the custom symbol 'EURUSD.C'
   Added 351183943 ticks to the history of the custom symbol 'EURUSD.C' in 269890 ms
   ...
   Requested 351183943 ticks to download tick history for the symbol 'EURUSD.C'
   The tick history for the 'EURUSD.C' symbol is received in the amount of 351183943 t
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.20 21:18:12.010

   The last 20 ticks for the custom symbol 'EURUSD.C':
     351183923th Tick: 2024.06.20 21:17:46.380 Ask=1.07124 Bid=1.07124 (Info tick)
     351183924th Tick: 2024.06.20 21:17:47.779 Ask=1.07125 Bid=1.07125 (Info tick)
     351183925th Tick: 2024.06.20 21:17:48.584 Ask=1.07124 Bid=1.07124 (Info tick)
     351183926th Tick: 2024.06.20 21:17:49.481 Ask=1.07125 Bid=1.07125 (Info tick)
     351183927th Tick: 2024.06.20 21:17:49.985 Ask=1.07122 Bid=1.07122 (Info tick)
     351183928th Tick: 2024.06.20 21:17:50.482 Ask=1.07124 Bid=1.07124 (Info tick)
     351183929th Tick: 2024.06.20 21:17:51.584 Ask=1.07123 Bid=1.07123 (Info tick)
     351183930th Tick: 2024.06.20 21:17:52.786 Ask=1.07124 Bid=1.07124 (Info tick)
     351183931th Tick: 2024.06.20 21:17:53.487 Ask=1.07125 Bid=1.07125 (Info tick)
     351183932th Tick: 2024.06.20 21:17:53.989 Ask=1.07126 Bid=1.07126 (Info tick)
     351183933th Tick: 2024.06.20 21:17:55.789 Ask=1.07125 Bid=1.07125 (Info tick)
     351183934th Tick: 2024.06.20 21:17:58.495 Ask=1.07126 Bid=1.07126 (Info tick)
     351183935th Tick: 2024.06.20 21:18:00.102 Ask=1.07126 Bid=1.07126 (Info tick)
     351183936th Tick: 2024.06.20 21:18:00.698 Ask=1.07129 Bid=1.07129 (Info tick)
     351183937th Tick: 2024.06.20 21:18:03.699 Ask=1.07129 Bid=1.07129 (Info tick)
     351183938th Tick: 2024.06.20 21:18:04.699 Ask=1.07128 Bid=1.07128 (Info tick)
     351183939th Tick: 2024.06.20 21:18:05.901 Ask=1.07129 Bid=1.07129 (Info tick)
     351183940th Tick: 2024.06.20 21:18:07.606 Ask=1.07128 Bid=1.07128 (Info tick)
     351183941th Tick: 2024.06.20 21:18:11.512 Ask=1.07127 Bid=1.07127 (Info tick)
     351183942th Tick: 2024.06.20 21:18:12.010 Ask=1.07126 Bid=1.07126 (Info tick)
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizad
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 53
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Obtiene el número indicado de ticks en la matriz                 |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- notificamos sobre el inicio de la carga de datos históricos
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count, sy

//--- haceremos tres intentos para obtener los ticks
   int attempts=0;
   while(attempts<3)
     {
      //--- medimos la hora de inicio antes de obtener los ticks
      uint start=GetTickCount();
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- solicitamos la historia de ticks desde el momento 1970.01.01 00:00.001 (pa
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- mostramos la información sobre el número de ticks y el tiempo invertido
         PrintFormat("The tick history for the '%s' symbol is received in the amount o

         //--- si la historia de ticks está sincronizada, el código de error será igua
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d ms.
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- calculamos los intentos
      attempts++;
      //--- pausa de 1 segundo a la espera de que finalice la sincronización de la bas
      Sleep(1000);
     }
//--- no ha sido posible copiar los ticks en 3 intentos
   return(false);
  }
//+------------------------------------------------------------------+
//| retorna la descripción de tipo string del tick                   |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|TIME_MINUTES

//--- comprobamos las banderas de tick
   bool buy_tick   = ((tick.flags &TICK_FLAG_BUY)    == TICK_FLAG_BUY);
   bool sell_tick  = ((tick.flags &TICK_FLAG_SELL)   == TICK_FLAG_SELL);
   bool ask_tick   = ((tick.flags &TICK_FLAG_ASK)    == TICK_FLAG_ASK);
   bool bid_tick   = ((tick.flags &TICK_FLAG_BID)    == TICK_FLAG_BID);
   bool last_tick  = ((tick.flags &TICK_FLAG_LAST)   == TICK_FLAG_LAST);
   bool volume_tick= ((tick.flags &TICK_FLAG_VOLUME)== TICK_FLAG_VOLUME);

//--- comprobamos primero si hay banderas comerciales en el tick (para CustomTicksAdd
   if(buy_tick || sell_tick)
     {
      //--- formamos la salida para el tick comercial
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last, tick
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last, tick
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- para la información del tick, formamos la salida un poco diferente
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- retornamos la descripción del tick
   return(desc);
  }
```

Vea también

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

CustomTicksDelete

Elimina todos los ticks en el intervalo temporal indicado de la historia de precio del instrumento personalizado.

```
int  CustomTicksDelete(
   const string     symbol,            // nombre del símbolo
   long             from_msc,          // desde qué fecha
   long             to_msc             // hasta qué fecha
   );
```

Parámetros

symbol

[in] Nombre del instrumento personalizado.

from_msc

[in] Hora del primer tick en la historia de precio del diapasón indicado, que debe ser eliminado. Hora en milisegundos desde el 01.01.1970.

to_msc

[in] Hora del último tick en la historia de precio del diapasón indicado, que debe ser eliminado. Hora en milisegundos desde el 01.01.1970.

Valor devuelto

Número de ticks eliminados o bien -1 en caso de error.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                            CustomTicksDelete.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // nombre del símbolo personalizad
#define   CUSTOM_SYMBOL_PATH     "Forex"           // nombre del grupo en el que se cr
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // nombre del símbolo a partir del

#define   DATATICKS_TO_COPY      UINT_MAX          // cantidad de ticks copiados
#define   DATATICKS_TO_DELETE    10                // cantidad de ticks a eliminar
#define   DATATICKS_TO_PRINT     20                // cantidad de ticks a mostrar en e

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- obtenemos en la matriz MqlTick los datos de los ticks del símbolo estándar
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;

//--- imprimimos la hora del primer y el último tick obtenidos del símbolo estándar
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- imprimimos en el registro DATATATICKS_TO_PRINT los últimos ticks del símbolo est
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':", DATATICKS_TO_PRINT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- añadimos el símbolo personalizado a la ventana MarketWatch (observación de merca
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- añadimos a la historia de precios del símbolo personalizado los datos de la mat
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'", ar
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms", add

//--- obtenemos en la matriz MqlTick los datos de los ticks del símbolo personalizado
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- imprimimos la hora del primer y el último tick obtenidos del símbolo personaliza
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- imprimimos en el registro DATATATICKS_TO_PRINT los últimos ticks del símbolo per
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':", DATATICKS_TO_PRINT,
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- obtenemos de la historia la hora del tick en milisegundos a partir de la cual bo
   long time_from=array[total-DATATICKS_TO_DELETE-1].time_msc;

//--- borramos DATATATICKS_TO_DELETE el rango de los últimos ticks del símbolo persona
   Print("...");
   start=GetTickCount();
   PrintFormat("Start deleting %u ticks in the history of the custom symbol '%s'", DAT
   int deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, time_from, array[total-2].time_ms
   PrintFormat("Deleted %u ticks in the history of the custom symbol '%s' in %u ms", d

//--- obtenemos en la matriz MqlTick los datos de los ticks del símbolo personalizado
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- imprimimos la hora del primer y el último tick del símbolo personalizado con el
   total=(int)array.Size();
   PrintFormat("Time of the first tick from the changed history: %s.%03u, Time of the
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- imprimimos en el registro DATATATICKS_TO_PRINT los últimos ticks del símbolo per
   PrintFormat("\nThe last %d ticks of custom symbol '%s' with modified history:", DAT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Sleep(16);
       //--- al presionar Supr, eliminamos el símbolo personalizado creado y sus datos
       if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
         {
          //--- eliminamos los datos de las barras
          int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
          if(deleted>0)
             PrintFormat("%d history bars of the custom symbol '%s' were successfully d

          //--- eliminamos los datos de ticks
          deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
          if(deleted>0)
             PrintFormat("%d history ticks of the custom symbol '%s' were successfully

          //--- eliminamos el símbolo
          if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
             PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
          break;
         }
      }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   Requested 4294967295 ticks to get tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351199027 tic
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 10:10:40.392

   The last 20 ticks for the standard symbol 'EURUSD':
     351199007th Tick: 2024.06.21 10:10:23.045 Bid=1.07032 (Info tick)
     351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
     351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 (Info tick)
     351199010th Tick: 2024.06.21 10:10:25.146 Bid=1.07032 (Info tick)
     351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
     351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
     351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
     351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
     351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
     351199016th Tick: 2024.06.21 10:10:30.258 Bid=1.07036 (Info tick)
     351199017th Tick: 2024.06.21 10:10:30.872 Ask=1.07035 Bid=1.07035 (Info tick)
     351199018th Tick: 2024.06.21 10:10:31.358 Ask=1.07036 (Info tick)
     351199019th Tick: 2024.06.21 10:10:31.859 Ask=1.07037 Bid=1.07037 (Info tick)
     351199020th Tick: 2024.06.21 10:10:32.377 Ask=1.07039 Bid=1.07039 (Info tick)
     351199021th Tick: 2024.06.21 10:10:32.962 Ask=1.0704 Bid=1.0704 (Info tick)
     351199022th Tick: 2024.06.21 10:10:33.961 Ask=1.07039 Bid=1.07039 (Info tick)
     351199023th Tick: 2024.06.21 10:10:34.667 Ask=1.0704 (Info tick)
     351199024th Tick: 2024.06.21 10:10:35.170 Bid=1.0704 (Info tick)
     351199025th Tick: 2024.06.21 10:10:38.266 Ask=1.07041 Bid=1.07041 (Info tick)
```

© 2000-2025, MetaQuotes Ltd.

```
  351199026th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
...
Start of adding 351199027 ticks to the history of the custom symbol 'EURUSD.C'
Added 351199027 ticks to the history of the custom symbol 'EURUSD.C' in 261594 ms
...
Requested 351199027 ticks to get tick history for the symbol 'EURUSD.C'
The tick history for the 'EURUSD.C' symbol is received in the amount of 351199027 t
First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 10:10:40.392

The last 20 ticks for the custom symbol 'EURUSD.C':
  351199007th Tick: 2024.06.21 10:10:23.045 Ask=1.07032 Bid=1.07032 (Info tick)
  351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
  351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 Bid=1.07032 (Info tick)
  351199010th Tick: 2024.06.21 10:10:25.146 Ask=1.07032 Bid=1.07032 (Info tick)
  351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
  351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
  351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
  351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
  351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
  351199016th Tick: 2024.06.21 10:10:30.258 Ask=1.07036 Bid=1.07036 (Info tick)
  351199017th Tick: 2024.06.21 10:10:30.872 Ask=1.07035 Bid=1.07035 (Info tick)
  351199018th Tick: 2024.06.21 10:10:31.358 Ask=1.07036 Bid=1.07036 (Info tick)
  351199019th Tick: 2024.06.21 10:10:31.859 Ask=1.07037 Bid=1.07037 (Info tick)
  351199020th Tick: 2024.06.21 10:10:32.377 Ask=1.07039 Bid=1.07039 (Info tick)
  351199021th Tick: 2024.06.21 10:10:32.962 Ask=1.0704 Bid=1.0704 (Info tick)
  351199022th Tick: 2024.06.21 10:10:33.961 Ask=1.07039 Bid=1.07039 (Info tick)
  351199023th Tick: 2024.06.21 10:10:34.667 Ask=1.0704 Bid=1.0704 (Info tick)
  351199024th Tick: 2024.06.21 10:10:35.170 Ask=1.0704 Bid=1.0704 (Info tick)
  351199025th Tick: 2024.06.21 10:10:38.266 Ask=1.07041 Bid=1.07041 (Info tick)
  351199026th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
...
Start deleting 10 ticks in the history of the custom symbol 'EURUSD.C'
Deleted 10 ticks in the history of the custom symbol 'EURUSD.C' in 188 ms
...
Requested 351199027 ticks to get tick history for the symbol 'EURUSD.C'
The tick history for the 'EURUSD.C' symbol is received in the amount of 351199017 t
Time of the first tick from the changed history: 2011.12.19 00:00:08.000, Time of t

The last 20 ticks of custom symbol 'EURUSD.C' with modified history:
  351198997th Tick: 2024.06.21 10:10:14.935 Ask=1.07036 Bid=1.07036 (Info tick)
  351198998th Tick: 2024.06.21 10:10:15.533 Ask=1.07035 Bid=1.07035 (Info tick)
  351198999th Tick: 2024.06.21 10:10:17.736 Ask=1.07036 Bid=1.07036 (Info tick)
  351199000th Tick: 2024.06.21 10:10:18.540 Ask=1.07037 Bid=1.07037 (Info tick)
  351199001th Tick: 2024.06.21 10:10:19.046 Ask=1.07038 Bid=1.07038 (Info tick)
  351199002th Tick: 2024.06.21 10:10:19.542 Ask=1.07036 Bid=1.07036 (Info tick)
  351199003th Tick: 2024.06.21 10:10:20.041 Ask=1.07035 Bid=1.07035 (Info tick)
  351199004th Tick: 2024.06.21 10:10:21.041 Ask=1.07035 Bid=1.07035 (Info tick)
  351199005th Tick: 2024.06.21 10:10:21.544 Ask=1.07032 Bid=1.07032 (Info tick)
  351199006th Tick: 2024.06.21 10:10:22.344 Ask=1.07032 Bid=1.07032 (Info tick)
```

© 2000-2025, MetaQuotes Ltd.

```
     351199007th Tick: 2024.06.21 10:10:23.045 Ask=1.07032 Bid=1.07032 (Info tick)
     351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
     351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 Bid=1.07032 (Info tick)
     351199010th Tick: 2024.06.21 10:10:25.146 Ask=1.07032 Bid=1.07032 (Info tick)
     351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
     351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
     351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
     351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
     351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
     351199016th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizad
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 53
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Obtiene el número indicado de ticks en la matriz                 |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- notificamos sobre el inicio de la carga de datos históricos
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count, sy

//--- haceremos tres intentos para obtener los ticks
   int attempts=0;
   while(attempts<3)
     {
      //--- medimos la hora de inicio antes de obtener los ticks
      uint start=GetTickCount();

      //--- solicitamos la historia de ticks desde el momento 1970.01.01 00:00.001 (pa
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- mostramos la información sobre el número de ticks y el tiempo invertido
         PrintFormat("The tick history for the '%s' symbol is received in the amount o

         //--- si la historia de ticks está sincronizada, el código de error será igua
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d ms.
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- calculamos los intentos
      attempts++;
      //--- pausa de 1 segundo a la espera de que finalice la sincronización de la bas
      Sleep(1000);
     }
//--- no ha sido posible copiar los ticks en 3 intentos
   return(false);
  }
//+------------------------------------------------------------------+
//| retorna la descripción de tipo string del tick                   |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|TIME_MINUTES
```

© 2000-2025, MetaQuotes Ltd.

```
//--- comprobamos las banderas de tick
   bool buy_tick   = ((tick.flags &TICK_FLAG_BUY)    == TICK_FLAG_BUY);
   bool sell_tick  = ((tick.flags &TICK_FLAG_SELL)   == TICK_FLAG_SELL);
   bool ask_tick   = ((tick.flags &TICK_FLAG_ASK)    == TICK_FLAG_ASK);
   bool bid_tick   = ((tick.flags &TICK_FLAG_BID)    == TICK_FLAG_BID);
   bool last_tick  = ((tick.flags &TICK_FLAG_LAST)   == TICK_FLAG_LAST);
   bool volume_tick= ((tick.flags &TICK_FLAG_VOLUME)== TICK_FLAG_VOLUME);

//--- comprobamos primero si hay banderas comerciales en el tick (para CustomTicksAdd
   if(buy_tick || sell_tick)
     {
      //--- formamos la salida para el tick comercial
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last, tick
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last, tick
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
      //--- para la información del tick, formamos la salida un poco diferente
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- retornamos la descripción del tick
   return(desc);
  }
```

Vea también

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

CustomTicksReplace

Sustituye totalmente la historia de precios del instrumento personalizado en el intervalo temporal indicado, con los datos de la matriz del tipo MqlTick.

```
int  CustomTicksReplace(
   const string     symbol,            // nombre del símbolo
   long             from_msc,          // desde qué fecha
   long             to_msc,            // hasta qué fecha
   const MqlTick&   ticks[],           // matriz con los datos de ticks que necesitamo
   uint             count=WHOLE_ARRAY  // número de elementos que se usarán de la mat
   );
```

Parámetros

symbol

[in] Nombre del instrumento personalizado.

from_msc

[in] Hora del primer tick en la historia de precio del diapasón indicado, que debe ser eliminado. Hora en milisegundos desde el 01.01.1970.

to_msc

[in] Hora del último tick en la historia de precio del diapasón indicado, que debe ser eliminado. Hora en milisegundos desde el 01.01.1970.

ticks[]

[in] Matriz de los datos de ticks del tipo MqlTick, organizados en orden temporalmente ascendente.

count=WHOLE_ARRAY

[in] Número de elementos de la matriz *ticks[]* que se usarán para la sustitución en el intervalo de tiempo indicado. El valor WHOLE_ARRAY indica que se debe usar todos los elementos de la matriz *ticks[]*.

Valor devuelto

Número de ticks actualizados o bien -1 en caso de error.

Nota

Puesto que en el flujo de cotizaciones no es raro que varios ticks tengan la misma hora con una precisión de milisegundos (la hora exacta del tick se guarda en el campo *time_msc* de la estructura MqlTick), la función CustomTicksReplace no realiza la clasificación automática de los elementos de la matriz *ticks[]* por hora. Por eso, la matriz de ticks debe ordenarse preliminarmente de forma temporalmente ascendente.

La sustitución de los ticks se realiza de forma consecutiva un día detrás otro hasta la hora indicada en *to_msc* o bien hasta el momento de aparición de un error. Primero se procesa el primer día del diapasón indicado, después el siguiente, y así sucesivamente. En cuanto se detecte una discordancia de la hora del tick con respecto al orden ascendente (no decreciente), el proceso de sustitución de ticks se interrumpirá de inmediato en el día actual. En este caso, además, los ticks

© 2000-2025, MetaQuotes Ltd.

de los días anteriores se sustituirán con éxito, mientras que el día actual (en el momento del tick incorrecto) y el resto de los días en el intervalo indicado permanecerán sin cambios.

Si en la matriz *ticks[]* no hay datos de ticks disponibles de algún día (en general, de una longitud cualquiera), después de aplicar los datos de ticks de `ticks[]` en la historia del instrumento personalizado se forma un "agujero", que se corresponde con los datos omitidos. En otras palabras, la llamada de CustomTicksReplace con ticks cortados en un intervalo concreto será equivalente a la eliminación de una parte de la historia de ticks, como si se llamara CustomTicksDelete con el intervalo "agujero".

Si no existen datos del intervalo de tiempo indicado en la base de ticks, CustomTicksReplace sencillamente añadirá a ella los ticks de la matriz ticks[].

La función CustomTicksReplace trabaja directamente con la base de datos de ticks.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                          CustomTicksReplace.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // nombre del símbolo personalizado
#define   CUSTOM_SYMBOL_PATH     "Forex"           // nombre del grupo en el que se cr
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // nombre del símbolo a partir del

#define   DATATICKS_TO_COPY      UINT_MAX          // cantidad de ticks copiados
#define   DATATICKS_TO_PRINT     20                // cantidad de ticks a mostrar en e

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el código de error al crear un símbolo personalizado
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- si el código de error no es 0 (el símbolo se ha creado con éxito) y no es 5304
   if(create!=0 && create!=5304)
      return;

//--- obtenemos en la matriz MqlTick los datos de los ticks del símbolo estándar
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- imprimimos la hora del primer y el último tick obtenidos del símbolo estándar
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- imprimimos en el registro DATATATICKS_TO_PRINT los últimos ticks del símbolo est
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':", DATATICKS_TO_PRINT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- añadimos el símbolo personalizado a la ventana MarketWatch (observación de merca
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- añadimos a la historia de precios del símbolo personalizado los datos de la mat
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'", a
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms", ad

//--- obtenemos en la matriz MqlTick los datos de los ticks del símbolo personalizado
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- imprimimos la hora del primer y el último tick obtenidos del símbolo personaliza
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- imprimimos en el registro DATATATICKS_TO_PRINT los últimos ticks del símbolo per
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':", DATATICKS_TO_PRINT,
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
```

© 2000-2025, MetaQuotes Ltd.

```
     }

//--- ahara cambiamos los valores Ask y Bid de los ticks en la matriz según la fórmula
   for(int i=0; i<total; i++)
     {
      array[i].ask = (array[i].ask !=0 ? 1.0 / array[i].ask : array[i].ask);
      array[i].bid = (array[i].bid !=0 ? 1.0 / array[i].bid : array[i].bid);
     }
   Print("\nNow the ticks are changed");

//--- sustituimos la historia de ticks del símbolo personalizado por los datos de la r
   Print("...");
   start=GetTickCount();
   PrintFormat("Start replacing %u changed ticks in the history of the custom symbol '
   int replaced=CustomTicksReplace(CUSTOM_SYMBOL_NAME, array[0].time_msc, array[total-
   PrintFormat("Replaced %u ticks in the history of the custom symbol '%s' in %u ms",

//--- obtenemos en la matriz MqlTick los datos de los ticks del símbolo personalizado
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- imprimimos la hora del primer y el último tick modificados del símbolo personali
   total=(int)array.Size();
   PrintFormat("First changed tick time: %s.%03u, Last changed tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- imprimimos en el registro DATATATICKS_TO_PRINT los últimos ticks modificados del
   PrintFormat("\nThe last %d changed ticks for the custom symbol '%s':", DATATICKS_TO
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Changed tick: %s", i, GetTickDescription(array[i]));
     }

//--- mostramos en el gráfico en el comentario la pista sobre las teclas de finalizaci
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- en un ciclo infinito esperamos que Esc o Del para la salida
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- al presionar Supr, eliminamos el símbolo personalizado creado y sus datos
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- eliminamos los datos de las barras
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
```

© 2000-2025, MetaQuotes Ltd.

```
            PrintFormat("%d history bars of the custom symbol '%s' were successfully c

         //--- eliminamos los datos de ticks
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- eliminamos el símbolo
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- antes de la salida, limpiamos el gráfico
   Comment("");
   /*
   resultado:
   Requested 4294967295 ticks to get tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351195822 tic
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 08:39:03.113

   The last 20 ticks for the standard symbol 'EURUSD':
     351195802th Tick: 2024.06.21 08:38:10.076 Ask=1.07194 (Info tick)
     351195803th Tick: 2024.06.21 08:38:13.162 Ask=1.07195 (Info tick)
     351195804th Tick: 2024.06.21 08:38:13.872 Bid=1.07195 (Info tick)
     351195805th Tick: 2024.06.21 08:38:14.866 Ask=1.07194 Bid=1.07194 (Info tick)
     351195806th Tick: 2024.06.21 08:38:17.374 Bid=1.07194 (Info tick)
     351195807th Tick: 2024.06.21 08:38:18.883 Bid=1.07194 (Info tick)
     351195808th Tick: 2024.06.21 08:38:19.771 Bid=1.07194 (Info tick)
     351195809th Tick: 2024.06.21 08:38:20.873 Ask=1.07195 Bid=1.07195 (Info tick)
     351195810th Tick: 2024.06.21 08:38:22.278 Ask=1.07196 Bid=1.07196 (Info tick)
     351195811th Tick: 2024.06.21 08:38:22.775 Bid=1.07196 (Info tick)
     351195812th Tick: 2024.06.21 08:38:23.477 Bid=1.07196 (Info tick)
     351195813th Tick: 2024.06.21 08:38:38.194 Ask=1.07197 (Info tick)
     351195814th Tick: 2024.06.21 08:38:38.789 Ask=1.07196 (Info tick)
     351195815th Tick: 2024.06.21 08:38:39.290 Ask=1.07197 (Info tick)
     351195816th Tick: 2024.06.21 08:38:43.695 Ask=1.07196 (Info tick)
     351195817th Tick: 2024.06.21 08:38:52.203 Ask=1.07195 Bid=1.07195 (Info tick)
     351195818th Tick: 2024.06.21 08:38:55.105 Ask=1.07196 Bid=1.07196 (Info tick)
     351195819th Tick: 2024.06.21 08:38:57.607 Ask=1.07195 Bid=1.07195 (Info tick)
     351195820th Tick: 2024.06.21 08:39:00.512 Ask=1.07196 Bid=1.07196 (Info tick)
     351195821th Tick: 2024.06.21 08:39:03.113 Ask=1.07195 Bid=1.07195 (Info tick)
   ...
   Start of adding 351195822 ticks to the history of the custom symbol 'EURUSD.C'
   Added 351195822 ticks to the history of the custom symbol 'EURUSD.C' in 349407 ms
   ...
   Requested 351195822 ticks to get tick history for the symbol 'EURUSD.C'
   The tick history for the 'EURUSD.C' symbol is received in the amount of 351195822 t
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 08:39:03.113
```

© 2000-2025, MetaQuotes Ltd.

```
The last 20 ticks for the custom symbol 'EURUSD.C':
  351195802th Tick: 2024.06.21 08:38:10.076 Ask=1.07194 Bid=1.07194 (Info tick)
  351195803th Tick: 2024.06.21 08:38:13.162 Ask=1.07195 Bid=1.07195 (Info tick)
  351195804th Tick: 2024.06.21 08:38:13.872 Ask=1.07195 Bid=1.07195 (Info tick)
  351195805th Tick: 2024.06.21 08:38:14.866 Ask=1.07194 Bid=1.07194 (Info tick)
  351195806th Tick: 2024.06.21 08:38:17.374 Ask=1.07194 Bid=1.07194 (Info tick)
  351195807th Tick: 2024.06.21 08:38:18.883 Ask=1.07194 Bid=1.07194 (Info tick)
  351195808th Tick: 2024.06.21 08:38:19.771 Ask=1.07194 Bid=1.07194 (Info tick)
  351195809th Tick: 2024.06.21 08:38:20.873 Ask=1.07195 Bid=1.07195 (Info tick)
  351195810th Tick: 2024.06.21 08:38:22.278 Ask=1.07196 Bid=1.07196 (Info tick)
  351195811th Tick: 2024.06.21 08:38:22.775 Ask=1.07196 Bid=1.07196 (Info tick)
  351195812th Tick: 2024.06.21 08:38:23.477 Ask=1.07196 Bid=1.07196 (Info tick)
  351195813th Tick: 2024.06.21 08:38:38.194 Ask=1.07197 Bid=1.07197 (Info tick)
  351195814th Tick: 2024.06.21 08:38:38.789 Ask=1.07196 Bid=1.07196 (Info tick)
  351195815th Tick: 2024.06.21 08:38:39.290 Ask=1.07197 Bid=1.07197 (Info tick)
  351195816th Tick: 2024.06.21 08:38:43.695 Ask=1.07196 Bid=1.07196 (Info tick)
  351195817th Tick: 2024.06.21 08:38:52.203 Ask=1.07195 Bid=1.07195 (Info tick)
  351195818th Tick: 2024.06.21 08:38:55.105 Ask=1.07196 Bid=1.07196 (Info tick)
  351195819th Tick: 2024.06.21 08:38:57.607 Ask=1.07195 Bid=1.07195 (Info tick)
  351195820th Tick: 2024.06.21 08:39:00.512 Ask=1.07196 Bid=1.07196 (Info tick)
  351195821th Tick: 2024.06.21 08:39:03.113 Ask=1.07195 Bid=1.07195 (Info tick)

Now the ticks are changed
...
Start replacing 351195822 changed ticks in the history of the custom symbol 'EURUSD
Replaced 351195822 ticks in the history of the custom symbol 'EURUSD.C' in 452266 m
...
Requested 351195822 ticks to get tick history for the symbol 'EURUSD.C'
The tick history for the 'EURUSD.C' symbol is received in the amount of 351195822 t
First changed tick time: 2011.12.19 00:00:08.000, Last changed tick time: 2024.06.2

The last 20 changed ticks for the custom symbol 'EURUSD.C':
  351195802th Changed tick: 2024.06.21 08:38:10.076 Ask=0.93289 Bid=0.93289 (Info t
  351195803th Changed tick: 2024.06.21 08:38:13.162 Ask=0.93288 Bid=0.93288 (Info t
  351195804th Changed tick: 2024.06.21 08:38:13.872 Ask=0.93288 Bid=0.93288 (Info t
  351195805th Changed tick: 2024.06.21 08:38:14.866 Ask=0.93289 Bid=0.93289 (Info t
  351195806th Changed tick: 2024.06.21 08:38:17.374 Ask=0.93289 Bid=0.93289 (Info t
  351195807th Changed tick: 2024.06.21 08:38:18.883 Ask=0.93289 Bid=0.93289 (Info t
  351195808th Changed tick: 2024.06.21 08:38:19.771 Ask=0.93289 Bid=0.93289 (Info t
  351195809th Changed tick: 2024.06.21 08:38:20.873 Ask=0.93288 Bid=0.93288 (Info t
  351195810th Changed tick: 2024.06.21 08:38:22.278 Ask=0.93287 Bid=0.93287 (Info t
  351195811th Changed tick: 2024.06.21 08:38:22.775 Ask=0.93287 Bid=0.93287 (Info t
  351195812th Changed tick: 2024.06.21 08:38:23.477 Ask=0.93287 Bid=0.93287 (Info t
  351195813th Changed tick: 2024.06.21 08:38:38.194 Ask=0.93286 Bid=0.93286 (Info t
  351195814th Changed tick: 2024.06.21 08:38:38.789 Ask=0.93287 Bid=0.93287 (Info t
  351195815th Changed tick: 2024.06.21 08:38:39.290 Ask=0.93286 Bid=0.93286 (Info t
  351195816th Changed tick: 2024.06.21 08:38:43.695 Ask=0.93287 Bid=0.93287 (Info t
  351195817th Changed tick: 2024.06.21 08:38:52.203 Ask=0.93288 Bid=0.93288 (Info t
```

© 2000-2025, MetaQuotes Ltd.

```
     351195818th Changed tick: 2024.06.21 08:38:55.105 Ask=0.93287 Bid=0.93287 (Info t
     351195819th Changed tick: 2024.06.21 08:38:57.607 Ask=0.93288 Bid=0.93288 (Info t
     351195820th Changed tick: 2024.06.21 08:39:00.512 Ask=0.93287 Bid=0.93287 (Info t
     351195821th Changed tick: 2024.06.21 08:39:03.113 Ask=0.93288 Bid=0.93288 (Info t
   */
  }
//+------------------------------------------------------------------+
//| Crea un símbolo personalizado, devuelve el código de error       |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- determinamos el nombre del símbolo a partir del cual se creará uno personalizad
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- si no hemos podido crear el símbolo personalizado, y no se trata de un error 53
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- con éxito
   return(error);
  }
//+------------------------------------------------------------------+
//| Elimina el símbolo personalizado                                 |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- ocultamos el símbolo de la ventana de Observación de mercado
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- si no se ha podido eliminar el símbolo personalizado, informaremos de ello en el
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- con éxito
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Obtiene el número indicado de ticks en la matriz                 |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- notificamos sobre el inicio de la carga de datos históricos
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count, sy

//--- haceremos tres intentos para obtener los ticks
   int attempts=0;
   while(attempts<3)
     {
      //--- medimos la hora de inicio antes de obtener los ticks
      uint start=GetTickCount();

      //--- solicitamos la historia de ticks desde el momento 1970.01.01 00:00.001 (pa
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- mostramos la información sobre el número de ticks y el tiempo invertido
         PrintFormat("The tick history for the '%s' symbol is received in the amount o

         //--- si la historia de ticks está sincronizada, el código de error será igua
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d ms.
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- calculamos los intentos
      attempts++;
      //--- pausa de 1 segundo a la espera de que finalice la sincronización de la bas
      Sleep(1000);
     }
//--- no ha sido posible copiar los ticks en 3 intentos
   return(false);
  }
//+------------------------------------------------------------------+
//| retorna la descripción de tipo string del tick                   |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|TIME_MINUTES

//--- comprobamos las banderas de tick
   bool buy_tick   = ((tick.flags &TICK_FLAG_BUY)   == TICK_FLAG_BUY);
   bool sell_tick  = ((tick.flags &TICK_FLAG_SELL)  == TICK_FLAG_SELL);
   bool ask_tick   = ((tick.flags &TICK_FLAG_ASK)   == TICK_FLAG_ASK);
   bool bid_tick   = ((tick.flags &TICK_FLAG_BID)   == TICK_FLAG_BID);
```

© 2000-2025, MetaQuotes Ltd.

```
   bool last_tick  = ((tick.flags &TICK_FLAG_LAST)  == TICK_FLAG_LAST);
   bool volume_tick= ((tick.flags &TICK_FLAG_VOLUME)== TICK_FLAG_VOLUME);

//--- comprobamos primero si hay banderas comerciales en el tick (para CustomTicksAdd
   if(buy_tick || sell_tick)
     {
      //--- formamos la salida para el tick comercial
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last, tick
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last, tick
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
      //--- para la información del tick, formamos la salida un poco diferente
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- retornamos la descripción del tick
   return(desc);
  }
```

Vea también

CustomRatesDelete, CustomRatesUpdate, CustomTicksDelete, CopyTicks, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

CustomBookAdd

Transmite el estado de la profundidad de mercado de un instrumento personalizado. La función permite emitir la profundidad de mercado como si llegase desde el servidor del bróker.

```
bool  CustomBookAdd(
   const string        symbol,            // nombre del símbolo
   const MqlBookInfo&  books[]            // matriz con las descripciones de los eleme
   uint                count=WHOLE_ARRAY  // número de elementos a utilizar
   );
```

Parámetros

symbol

[in] Nombre del símbolo personalizado.

books[]

[in] Matriz de datos del tipo MqlBookInfo que describen al completo el estado de la profundidad de mercado — todas las solicitudes de compra y venta. El estado transmitido de la profundidad de mercado sustituye por completo al anterior.

count=WHOLE_ARRAY

[in] Número de elementos de la matriz *books* que deberá ser transmitido a la función. Por defecto, se usa toda la matriz.

Valor retornado

true en el caso de éxito, de lo contrario, false. Para obtener información sobre el error, necesitamos llamar la función GetLastError().

Observación

La función CustomBookAdd funciona solo con los símbolos personalizados para los que está abierta la profundidad de mercado, a través de la plataforma o la funciónMarketBookAdd.

Al añadirse información a la profundidad de mercado, los precios Bid y Ask del instrumento no se actualizan. Usted deberá controlar por su propia cuenta el cambio de los mejores precios e incluir los ticks con la ayuda de CustomTicksAdd.

Se comprueba que los datos transmitidos sean correctos: para cada elemento hay que especificar el tipo, el precio y el volumen. MqlBookInfo.volume y MqlBookInfo.volume_real no deberán ser cero o negativos, si ambos volúmenes son negativos, se considerará un error. Podemos especificar uno o ambos volúmenes: se tomará el indicado o el positivo:

```
volume=-1 && volume_real=2 — se utilizará volume_real=2,
a volume=3 && volume_real=0 — se utilizará volume=3.
```

El volumen con precisión aumentada MqlBookInfo.volume_real tiene prioridad en comparación con el habitual MqlBookInfo.volume. Si para un elemento de la profundidad de mercado se han indicado ambos, se usará volume_real.

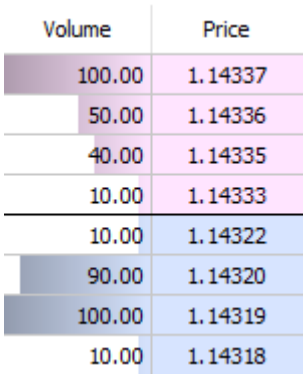
El orden de secuencia de los elementos de MqlBookInfo en la matriz *books* no importa. Al guardar los datos, el terminal los clasifica por precio independientemente.

© 2000-2025, MetaQuotes Ltd.

Al guardar los datos se comprueba el parámetro "Profundidad del mercado" (SYMBOL_TICKS_BOOKDEPTH) del instrumento personalizado que recibe. Si el número de solicitudes de venta en la profundidad de mercado transmitida supera este valor, los niveles sobrantes serán descartados. De una forma análoga sucederá con las solicitudes de compra.

Ejemplo de rellanado de la matriz *books*:

Estado de la profundidad de mercado	Rellenado de books[]

	books[0].type=BOOK_TYPE_SELL;
	books[0].price=1.14337;
	books[0].volume=100;
	books[1].type=BOOK_TYPE_SELL;
	books[1].price=1.14330;
	books[1].volume=50;
	books[2].type=BOOK_TYPE_SELL;
	books[2].price=1.14335;
	books[2].volume=40;
	books[3].type=BOOK_TYPE_SELL;
	books[3].price=1.14333;
	books[3].volume=10;
	books[4].type=BOOK_TYPE_BUY;
	books[4].price=1.14322;
	books[4].volume=10;
	books[5].type=BOOK_TYPE_BUY;
	books[5].price=1.14320;
	books[5].volume=90;
	books[6].type=BOOK_TYPE_BUY;
	books[6].price=1.14319;
	books[6].volume=100;
	books[7].type=BOOK_TYPE_BUY;
	books[7].price=1.14318;
	books[7].volume=10;

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- activamos la profundidad de mercado para el instrumento del que tomaremos los da
   MarketBookAdd(Symbol());
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//| Tick function                                                    |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   MqlTick ticks[];
   ArrayResize(ticks,1);
//--- copiamos los precios actuales desde un instrumento habitual a uno personalizado
   if(SymbolInfoTick(Symbol(),ticks[0]))
     {
      string symbol_name=Symbol()+".SYN";
      CustomTicksAdd(symbol_name,ticks);
     }
  }
//+------------------------------------------------------------------+
//| Book function                                                    |
//+------------------------------------------------------------------+
void OnBookEvent(const string &book_symbol)
  {
//--- copiamos el estado actual de la profundidad de mercado desde un instrumento habi
   if(book_symbol==Symbol())
     {
      MqlBookInfo book_array[];
      if(MarketBookGet(Symbol(),book_array))
        {
         string symbol_name=Symbol()+".SYN";
         CustomBookAdd(symbol_name,book_array);
        }
     }
  }
//+------------------------------------------------------------------+
```

Ver también

MarketBookAdd, CustomTicksAdd, OnBookEvent

© 2000-2025, MetaQuotes Ltd.

Operaciones con gráficos

Las funciones para establecer las propiedades del gráfico (ChartSetInteger, ChartSetDouble, ChartSetString) son asincrónicas y sirven para enviar al gráfico un comando de cambio. Si dichas funciones se ejecutan con éxito, el comando entra en la cola general de eventos del gráfico. El cambio de propiedades del gráfico se realiza durante el procesamiento de la cola de eventos de dicho gráfico.

Por este motivo, no cabe esperar la actualización inmediata del gráfico después de llamar funciones asincrónicas. Para actualizar de manera forzosa el aspecto externo y las propiedades del gráfico, use la función ChartRedraw().

Función	Acción
ChartApplyTemplate	Aplica al gráfico especificado una plantilla desde un archivo especificado
ChartSaveTemplate	Guarda los ajustes actuales del gráfico en una plantilla con el nombre especificado
ChartWindowFind	Devuelve el número de una subventana en la que se encuentra el indicador
ChartTimePriceToXY	Convierte las coordenadas del gráfico desde la representación hora/precio a las coordinadas en el eje X y Y
ChartXYToTimePrice	Convierte las coordenadas X y Y del gráfico a los valores hora y precio
ChartOpen	Abre un gráfico nuevo con un símbolo y período especificados
ChartClose	Cierra un gráfico especificado
ChartFirst	Devuelve el identificador del primer gráfico del terminal de cliente
ChartNext	Devuelve el identificador del gráfico que sigue después del especificado
ChartSymbol	Devuelve el nombre del símbolo del gráfico especificado
ChartPeriod	Devuelve el valor del período del gráfico especificado
ChartRedraw	Activa el redibujo forzado de un gráfico especificado
ChartSetDouble	Establece el valor del tipo double de una propiedad correspondiente del gráfico especificado
ChartSetInteger	Establece el valor del tipo entero (datetime, int, color, bool o char) de una propiedad correspondiente del gráfico especificado
ChartSetString	Establece el valor del tipo string de una propiedad correspondiente del gráfico especificado
ChartGetDouble	Devuelve el valor de una propiedad correspondiente del gráfico especificado
ChartGetInteger	Devuelve el valor del tipo entero de una propiedad correspondiente del gráfico especificado

© 2000-2025, MetaQuotes Ltd.

Función	Acción
ChartGetString	Devuelve el valor literal de una propiedad correspondiente del gráfico especificado
ChartNavigate	Desplaza el gráfico especificado a una cantidad de barras especificada respecto a la posición del gráfico indicada
ChartID	Devuelve el identificador del gráfico corriente
ChartIndicatorAdd	Añade un indicador con el manejador (handle) especificado a la ventana del gráfico especificado
ChartIndicatorDelete	Quita el indicador con el nombre especificado de la ventana del gráfico especificada
ChartIndicatorGet	Devuelve el manejador del indicador con el nombre corto especificado en la ventana del gráfico especificada
ChartIndicatorName	Devuelve el nombre breve del indicador según su número en la lista de indicadores en la determinada ventana del gráfico
ChartIndicatorsTotal	Devuelve el número de todos los indicadores vinculados con la determinada ventana del gráfico
ChartWindowOnDropped	Devuelve el número de la subventana del gráfico a la que el Asesor Experto, script, objeto o indicador ha sido arrastrado con el ratón
ChartPriceOnDropped	Devuelve la coordinada de precios que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón
ChartTimeOnDropped	Devuelve la coordinada de tiempo que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón
ChartXOnDropped	Devuelve la coordinada del eje de X que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón
ChartYOnDropped	Devuelve la coordinada del eje de Y que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón
ChartSetSymbolPeriod	Cambia el valor del símbolo y período del gráfico especificado
ChartScreenShot	Hace un screenshot del gráfico especificado en formato GIF, PNG o BMP, dependiendo de la extensión especificada

© 2000-2025, MetaQuotes Ltd.

ChartApplyTemplate

Aplica al gráfico una plantilla especificada. El comando se añade a la cola de los mensajes del gráfico y se ejecuta sólo después del procesamiento de todos los comandos anteriores. Si el comando se coloca con éxito a la cola del gráfico, devuelve true, de lo contrario, devuelve false.

```
bool  ChartApplyTemplate(
   long          chart_id,     // identificador del gráfico
   const string  filename      // nombre del archivo con la plantilla
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

filename

[in] Nombre del archivo que contiene la plantilla.

Valor devuelto

Si el comando se coloca con éxito a la cola del gráfico, devuelve true, de lo contrario, devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Si a través de esta función desde el Asesor Experto se cargará una plantilla nueva en el gráfico al que este Asesor Experto está adjuntado, entonces este Asesor Experto será descargado y no podrá continuar su trabajo.

Al aplicar la plantilla al gráfico, el derecho al trading puede ser restringido por razones de seguridad:

> **Los derechos al trading no pueden ser ampliados cuando el EA se inicia por la aplicación de las plantillas mediante la función ChartApplyTemplate().**

Si el programa mql5 que llama a la función ChartApplyTemplate() no tiene los derechos al trading, el EA cargado mediante la plantilla tampoco tendrá los derechos al trading independientemente de los ajustes de la plantilla.

Si el programa mql5 que llama a la función ChartApplyTemplate() tiene los derechos al trading y en los ajustes de la plantilla no hay estos derechos, el EA cargado mediante la plantilla no tendrá los derechos al trading.

Uso de plantillas

Los recursos del lenguaje MQL5 permiten establecer varias propiedades del gráfico, comprendido entre ellas los colores, utilizando la función ChartSetInteger() :

- Color del fondo del gráfico;
- Color de ejes, escala y línea OHLC;
- Color de cuadrícula;
- Color de volúmenes y niveles de apertura de posiciones;
- Color de la barra arriba, sombra y borde del cuerpo de vela alcista;

© 2000-2025, MetaQuotes Ltd.

- Color de la barra abajo, sombra y borde del cuerpo de vela bajista;
- Color de la línea del gráfico y velas japonesas "Doji";
- Color del cuerpo de vela alcista;
- Color del cuerpo de vela bajista;
- Color de la línea del precio Bid;
- Color de la línea del precio Ask;
- Color de la línea del precio de la última transacción realizada (Last);
- Color de niveles de órdenes Stop (Stop Loss y Take Profit).

Además, en el gráfico pueden haber varios objetos gráficos e indicadores. Será suficiente configurar una vez la apariencia del gráfico, dotándolo con todos los indicadores necesarios, y guardarlo como una plantilla para luego poder aplicar esta plantilla a cualquier gráfico.

La función ChartApplyTemplate() sirve para usar las plantillas guardadas anteriormente. Esta función se puede utilizar en cualquier programa mql5. La ruta del archivo en el que se encuentra la plantilla se pasa con el segundo parámetro de la función ChartApplyTemplate(). La ruta del archivo en el que se encuentra la plantilla se pasa. La búsqueda del archivo de una plantilla se realiza según las siguientes reglas:

- si la ruta se empieza con la barra inversa separadora "\" (se escribe "\\"), entonces la plantilla se busca según la ruta catálogo_de_datos_del_terminal\MQL5,
- si no hay ninguna barra inversa, la plantilla se busca respecto a la ubicación del archivo ejecutable EX5 en el que se realiza la llamada a la función ChartApplyTemplate();
- si la plantilla no ha sido encontrada en las primeras dos opciones, la búsqueda continúa en la carpeta directorio_del_terminal\Profiles\Templates\.

Aquí directorio_del_terminal significa la carpeta desde la que ha sido iniciado el terminal de cliente MetaTrader 5, mientras que catálogo_de_datos_del_terminal significa la carpeta en la que se guardan los archivos editables y su ubicación puede depender del tipo del sistema operativo, nombre del usuario y las configuraciones de seguridad del ordenador. Por lo general, se trata de diferentes carpetas, aunque en algunas ocasiones pueden coincidir.

La ubicación de las carpetas catálogo_de_datos_del_terminal y directorio_del_terminal se puede averiguar a través de la función TerminalInfoString().

```
//--- carpeta desde la que ha sido iniciado el terminal
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
   Print("Directorio del terminal:",terminal_path);
//--- carpeta de datos del terminal que contiene la carpeta MQL5 con los EAs e indicad
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   Print("Carpeta de datos del terminal:",terminal_data_path);
```

Ejemplos de entrada:

```
//--- buscamos la plantilla en la carpeta catálogo_de_datos_del_terminal\MQL5\
ChartApplyTemplate(0,"\\first_template.tpl"))
//--- buscamos la plantilla en la carpeta catálogo_del_archivo_ejecutable_EX5\, luego
ChartApplyTemplate(0,"second_template.tpl"))
//--- buscamos la plantilla en la carpeta catálogo_del_archivo_ejecutable_EX5\My_templ
ChartApplyTemplate(0,"My_templates\\third_template.tpl"))
```

Las plantillas no pertenecen a los recursos, y no se puede insertarlos en un archivo ejecutable EX5.

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ejemplo de aplicación de la plantilla ubicada en la carpeta \MQL5\Files
   if(FileIsExist("my_template.tpl"))
     {
      Print("Plantilla my_template.tpl encontrada en la carpeta \Files'");
      //--- aplicamos la plantilla encontrada
      if(ChartApplyTemplate(0,"\\Files\\my_template.tpl"))
        {
         Print("La plantilla 'my_template.tpl' ha sido aplicada con éxito");
         //--- redibujamos forzosamente el gráfico para la visualización rápida de los
         ChartRedraw();
        }
      else
         Print("Fallo al aplicar la plantilla 'my_template.tpl', error ",GetLastError
     }
   else
     {
      Print("Archivo 'my_template.tpl' no encontrado en la carpeta "
            +TerminalInfoString(TERMINAL_PATH)+"\\MQL5\\Files");
     }
  }
```

Véase también

Recursos

© 2000-2025, MetaQuotes Ltd.

ChartSaveTemplate

Guarda los ajustes actuales del gráfico en una plantilla con el nombre especificado.

```
bool  ChartSaveTemplate(
   long          chart_id,     // identificador del gráfico
   const string  filename      // nombre del archivo para guardar la plantilla
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

filename

[in] Nombre del archivo para guardar la plantilla. La extensión ".tpl" se añade al nombre del archivo de forma automática, no hace falta ponerla. La plantilla se guarda en la carpeta **directorio_del_terminal**\Profiles\Templates\, y puede ser utilizada también para la aplicación manual en el terminal. Si ya existe una plantilla con el mismo nombre, entonces su contenido será sobrescrito de nuevo.

Valor devuelto

En caso del éxito la función devuelve true, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Una plantilla permite guardar los ajustes del gráfico con todos los indicadores y objetos gráficos aplicados a él para que luego el usuario pueda aplicarla al otro gráfico.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                      Test_ChartSaveTemplate.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- input parameters
input string               symbol="GBPUSD";  // símbolo del gráfico nuevo
input ENUM_TIMEFRAMES      period=PERIOD_H3; // período de tiempo del gráfico nuevo
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- primero adjuntamos al gráfico los indicadores
   int handle;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- preparamos los indicadores para el uso
   if(!PrepareZigzag(NULL,0,handle)) return; // fallo, entonces salimos
//--- colocamos el indicador en el gráfico actual, pero... en otra ventana :)
   if(!ChartIndicatorAdd(0,1,handle))
     {
      PrintFormat("Fallo al adjuntar al gráfico %s/%s el indicador con el manejador=%d
                  _Symbol,
                  EnumToString(_Period),
                  handle,
                  GetLastError());
      //--- finalizamos anticipadamente el trabajo del programa
      return;
     }
//--- actualizamos el gráfico para ver el indicador colocado
   ChartRedraw();
//--- encontraremos los dos últimos virajes del zigzag
   double two_values[];
   datetime two_times[];
   if(!GetLastTwoFractures(two_values,two_times,handle))
     {
      PrintFormat("No se ha podido encontrar los dos últimos virajes en el indicador Z
      //--- finalizamos anticipadamente el trabajo del programa
      return;
     }
//--- ahora adjuntamos el canal de desviación estándar
   string channel="StdDeviation Channel";
   if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
     {
      PrintFormat("Fallo al crear el objeto %s. Código del error %d",
                  EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
      return;
     }
   else
     {
      //--- el canal está creado, definimos el segundo punto de apoyo
      ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
      //--- asignaremos al canal el texto de ayuda emergente
      ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demo from MQL5 Help");
      //--- actualizaremos el gráfico
      ChartRedraw();
     }
//--- guardaremos toda esta maravilla en la plantilla
   ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- abriremos nuevo gráfico y aplicaremos la plantilla guardada
   long new_chart=ChartOpen(symbol,period);
   //--- activaremos la opción de descripción emergente para los objetos gráficos
   ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
   if(new_chart!=0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- aplicaremos al gráfico nuevo la plantilla guardada
      ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
     }
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| crea el manejador del zigzag y asegura la disponibilidad de sus datos       |
//+------------------------------------------------------------------+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
  {
   ResetLastError();
//--- el indicador Zigzag debe ubicarse en la carpeta catálogo_de_datos_del_terminal\M
   h=iCustom(sym,tf,"Examples\\Zigzag");
   if(h==INVALID_HANDLE)
     {
      PrintFormat("%s: Fallo al crear el manejador del indicador Zigzag. Código del er
                  __FUNCTION__,GetLastError());
      return false;
     }
//--- cuando se crea el manejador del indicador, él necesita tiempo para calcular los
   int k=0; // número de intentos para conseguir la calculación del indicador
//--- esperamos el cálculo en el ciclo, hacemos una pausa de 50 millisegundos si el cá
   while(BarsCalculated(h)<=0)
     {
      k++;
      //--- mostramos el número de intentos
      PrintFormat("%s: k=%d",__FUNCTION__,k);
      //--- esperamos 50 millisegundos hasta que el indicador se calcule
      Sleep(50);
      //--- si han sido más de 100 intentos, algo va mal
      if(k>100)
        {
         //--- avisamos sobre el problema
         PrintFormat("No se ha podido calcular el indicador con %d intentos!");
         //--- finalizamos anticipadamente el trabajo del programa
         return false;
        }
     }
//--- todo está listo, el indicador está creado y los valores están calculados
   return true;
  }
//+------------------------------------------------------------------+
//|  busca los dos últimos virajes del zigzag y coloca en los arrays        |
//+------------------------------------------------------------------+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
  {
   double values[];          // array para obtener los valores del zigzag
   datetime times[];         // array para obtener el tiempo
   int size=100;             // tamaño de arrays
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- copiamos los 100 últimos valores del indicador
   int copied=CopyBuffer(handle,0,0,size,values);
//--- comprobamos cuántos se han copiado
   if(copied<100)
     {
      PrintFormat("%s: No se han copiado %d valores del indicador con el manejador=%d.
                  __FUNCTION__,size,handle,GetLastError());
      return false;
     }
//--- definimos el orden de acceso al array como en una serie temporal
   ArraySetAsSeries(values,true);
//--- escribiremos aquí los números de las barras en las que se han encontrado los vi
   int positions[];
//--- fijaremos los tamaños de arrays
   ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//--- contadores
   int i=0,k=0;
//--- empezamos a buscar los virajes
   while(i<100)
     {
      double v=values[i];
      //--- los valores vacíos no nos interesan
      if(v!=0.0)
        {
         //--- recordamos el número de la barra
         positions[k]=i;
         //--- recordamos el valor del zigzag en el viraje
         get_values[k]=values[i];
         PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
         //--- aumentamos contadores
         k++;
         //--- si hemos encontrado dos virajes, rompemos el ciclo
         if(k>2) break;
        }
      i++;
     }
//--- definimos el orden de acceso a los arrays como en una serie temporal
   ArraySetAsSeries(times,true);   ArraySetAsSeries(get_times,true);
   if(CopyTime(_Symbol,_Period,0,size,times)<=0)
     {
      PrintFormat("%s: Fallo al copiar %d valores desde CopyTime(). Código del error %
                  __FUNCTION__,size,GetLastError());
      return false;
     }
//--- encontraremos el tiempo de apertura de las barras en las que han sido encontrad
   get_times[0]=times[positions[1]];// el penúltimo valor se escribirá como el primer
   get_times[1]=times[positions[2]];// el tercer valor desde el final será el segundo
   PrintFormat("%s: el primero=%s,  el segundo=%s",__FUNCTION__,TimeToString(get_times
```

© 2000-2025, MetaQuotes Ltd.

```
//--- todo ha salido bien
   return true;
  }
```

Véase también

ChartApplyTemplate(), Recursos

© 2000-2025, MetaQuotes Ltd.

ChartWindowFind

Devuelve el número de una subventana en la que se encuentra el indicador. Existen 2 variantes de la función.

1. La función busca en el gráfico especificado la subventana con el "nombre breve" del indicador (el nombre breve se muestra en la parte superior izquierda de la subventana), y en caso del éxito devuelve el número de subventana.

```
int  ChartWindowFind(
   long     chart_id,                  // identificador del gráfico
   string   indicator_shortname        // nombre breve del indicador, véase INDICATOR
```

2. La función debe invocarse desde el indicador personalizado y devuelve el número de la subventana en la que este indicador trabaja.

```
int  ChartWindowFind();
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

indicator_shortname

[in] Nombre breve del indicador.

Valor devuelto

Devuelve el número de subventana en caso del éxito. Cero significa la ventana principal del gráfico. En caso de fallo devuelve -1.

Nota

Si se llama a la segunda variante de la función (sin parámetros) desde un script o Asesor Experto, devuelve -1.

No se debe confundir el nombre breve del indicador con el nombre del archivo que se indica durante la creación del indicador por las funciones iCustom() y IndicatorCreate(). Si el nombre breve del indicador no se especifica de manera explícita, entonces durante la compilación ahí se indica el nombre del archivo con el código fuente del indicador.

Hay que formar correctamente el nombre breve del indicador que se escribe en la propiedad INDICATOR_SHORTNAME mediante la función IndicatorSetString(). Es recomendable que el nombre breve contenga los valores de los parámetros de entrada del indicador, puesto que en la función ChartIndicatorDelete() la identificación del indicador que se quita del gráfico se realiza precisamente por el nombre breve.

Ejemplo:

```
#property script_show_inputs
//--- input parameters
input string   shortName="MACD(12,26,9)";
//+------------------------------------------------------------------+
//|  devuelve el número de la ventana del gráfico con este indicador |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
  {
   int window=-1;
//---
   if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
     {
      //--- la función ha sido invocada desde el indicador, no hace falta el nombre
      window=ChartWindowFind();
     }
   else
     {
      //--- la función ha sido invocada desde el Asesor Experto o script
      window=ChartWindowFind(0,short_name);
      if(window==-1) Print(__FUNCTION__+"(): Error = ",GetLastError());
     }
//---
   return(window);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int window=GetIndicatorSubWindowNumber(0,shortName);
   if(window!=-1)
      Print("Indicador "+shortName+" se encuentra en la ventana #"+(string)window);
   else
      Print("Indicador "+shortName+" no ha sido encontrado. window = "+(string)window)
  }
```

Véase también

ObjectCreate(), ObjectFind()

© 2000-2025, MetaQuotes Ltd.

ChartTimePriceToXY

Convierte las coordenadas del gráfico desde la representación hora/precio a las coordinadas en el eje X y Y.

```
bool  ChartTimePriceToXY(
   long           chart_id,      // identificador del gráfico
   int            sub_window,    // número de subventana
   datetime       time,          // fecha/hora en el gráfico
   double         price,         // precio en el gráfico
   int&           x,             // coordinada X para la hora en el gráfico
   int&           y              // coordinada Y para el precio en el gráfico
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

sub_window

[in] Número de subventana del gráfico. 0 significa la ventana principal del gráfico.

time

[in] Valor de la hora en el gráfico para el cual se recibirá el valor en píxeles en el eje X. El inicio de las coordinadas se encuentra en la esquina superior izquierda de la ventana principal del gráfico.

price

[in] Valor del precio en el gráfico para el cual se recibirá el valor en píxeles en el eje Y. El inicio de las coordinadas se encuentra en la esquina superior izquierda de la ventana principal del gráfico.

x

[out] Variable en la que se recibirá la conversión de la hora a la coordinada X.

y

[out] Variable en la que se recibirá la conversión del precio a la coordinada Y.

Valor devuelto

Devuelve true en caso de la ejecución con éxito, de lo contrario - false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Ejemplo:

```
#define   BAR_NUMBER    0  // número de barra desde el que obtenemos el precio y el ti

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- copiamos los datos de una barra según el índice BAR_NUMBER
   MqlRates rates[]={};
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyRates(_Symbol, _Period, BAR_NUMBER, 1, rates)!=1)
     {
      PrintFormat("CopyRates() failed for bar %d. Error %d", BAR_NUMBER, GetLastError
      return;
     }

//--- convertimos el precio y el tiempo obtenidos en coordenadas de píxeles del gráfic
   int x=0, y=0;
   ResetLastError();
   if(!ChartTimePriceToXY(ChartID(), 0, rates[0].time, rates[0].close, x, y))
     {
      Print("ChartTimePriceToXY() failed. Error ", GetLastError());
      return;
     }

//--- imprimimos el resultado obtenido en el registro
   PrintFormat("For bar[%d] with opening time %s and price %.*f, the chart coordinates

   /*
   resultado:
   For bar[0] with opening time 2024.08.09 15:06 and price 1.27378, the chart coordina
   */
  }
```

Véase también

ChartXYToTimePrice()

© 2000-2025, MetaQuotes Ltd.

ChartXYToTimePrice

Convierte las coordenadas X y Y del gráfico a los valores hora y precio.

```
bool  ChartXYToTimePrice(
   long           chart_id,      // identificador del gráfico
   int            x,             // coordinada X en el gráfico
   int            y,             // coordinada Y en el gráfico
   int&           sub_window,    // número de subventana
   datetime&      time,          // fecha/hora en el gráfico
   double&        price          // precio en el gráfico
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

x

[in] Coordinada X.

y

[in] Coordinada Y.

sub_window

[out] Variable en la que será escrito el número de la subventana del gráfico. 0 significa la ventana principal del gráfico.

time

[out] Valor de la hora en el gráfico para el cual se recibirá el valor en píxeles en el eje X. El inicio de las coordinadas se encuentra en la esquina superior izquierda de la principal del gráfico.

price

[out] Valor del precio en el gráfico para el cual se recibirá el valor en píxeles en el eje Y. El inicio de las coordinadas se encuentra en la esquina superior izquierda de la principal del gráfico.

Valor devuelto

Devuelve true en caso de la ejecución con éxito, de lo contrario - false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- mostramos los parámetros del evento en el gráfico
   Comment(__FUNCTION__,": id=",id," lparam=",lparam," dparam=",dparam," sparam=",spar
//--- si se trata del evento del cliqueo sobre el gráfico
   if(id==CHARTEVENT_CLICK)
     {
      //--- preparamos las variables
      int      x     =(int)lparam;
      int      y     =(int)dparam;
      datetime dt    =0;
      double   price =0;
      int      window=0;
      //--- convertimos las coordinadas X y Y  en los términos fecha/hora
      if(ChartXYToTimePrice(0,x,y,window,dt,price))
        {
         PrintFormat("Window=%d X=%d  Y=%d  =>  Time=%s  Price=%G",window,x,y,TimeToSt
         //--- hacemos la conversión inversa: (X,Y) => (Time,Price)
         if(ChartTimePriceToXY(0,window,dt,price,x,y))
            PrintFormat("Time=%s  Price=%G  =>  X=%d  Y=%d",TimeToString(dt),price,x,y
         else
            Print("ChartTimePriceToXY return error code: ",GetLastError());
         //--- delete lines
         ObjectDelete(0,"V Line");
         ObjectDelete(0,"H Line");
         //--- create horizontal and vertical lines of the crosshair
         ObjectCreate(0,"H Line",OBJ_HLINE,window,dt,price);
         ObjectCreate(0,"V Line",OBJ_VLINE,window,dt,price);
         ChartRedraw(0);
        }
      else
         Print("ChartXYToTimePrice return error code: ",GetLastError());
      Print("+------------------------------------------------------------------+");
     }
  }
```

Véase también

ChartTimePriceToXY()

© 2000-2025, MetaQuotes Ltd.

ChartOpen

Abre un gráfico nuevo con un símbolo y período especificados

```
long  ChartOpen(
   string           symbol,     // nombre del símbolo
   ENUM_TIMEFRAMES  period      // período
   );
```

Parámetros

symbol

[in] Símbolo del gráfico. NULL significa el símbolo del gráfico corriente (al que está adjuntado Asesor Experto).

period

[in] Período del gráfico (período de tiempo). Puede adquirir uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 significa el período del gráfico corriente.

Valor devuelto

En caso del éxito la función devuelve el identificador del gráfico. De lo contrario devuelve 0.

Nota

El número máximo posible de gráficos abiertos a la vez en el terminal no puede superar el valor de CHARTS_MAX.

Ejemplo:

```
#define CHART_SYMBOL  NULL
#define CHART_PERIOD  PERIOD_CURRENT

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- establecemos el símbolo y el marco temporal para el nuevo gráfico
   string symbol=CHART_SYMBOL;
   if(symbol==NULL)
      symbol=Symbol();
   ENUM_TIMEFRAMES timeframe = (CHART_PERIOD==PERIOD_CURRENT ? Period() : CHART_PERIOI

//--- abrimos el nuevo gráfico con el símbolo y periodo establecidos
   long chart_id=ChartOpen(symbol, timeframe);
   if(chart_id==0)
     {
      Print("ChartOpen() failed. Error ", GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- imprimimos en el registro los parámetros del gráfico abierto
   PrintFormat("A new chart of the %s symbol has been opened with a period of %s and C
               symbol, StringSubstr(EnumToString(timeframe), 7), chart_id);
   /*
   resultado:
   A new chart of the GBPUSD symbol has been opened with a period of M1 and ChartID 13
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartFirst

Devuelve el indicador del primer gráfico del terminal de cliente.

```
long  ChartFirst();
```

Valor devuelto

Identificador del gráfico.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el identificador del primer gráfico abierto en el terminal de cliente
   long            chart_id= ChartFirst();

//--- según el identificador recibido, obtenemos adicionalmente el símbolo y el perioc
   string          symbol  = ChartSymbol(chart_id);
   ENUM_TIMEFRAMES period  = ChartPeriod(chart_id);

//--- mostramos la descripción del primer gráfico del terminal del cliente
   PrintFormat("ID of the first chart of the client terminal: %I64d, chart symbol: %s,
   /*
   resultado:
   ID of the first chart of the client terminal: 133246248352168440, chart symbol: EUI
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartNext

Devuelve el identificador del gráfico que sigue después del gráfico especificado.

```
long  ChartNext(
   long  chart_id      // identificador del gráfico
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 no significa el gráfico corriente. 0 significa "devolver el identificador del primer gráfico".

Valor devuelto

Identificador del gráfico. Si la lista de gráficos se ha terminado, la función devuelve -1.

Ejemplo:

```
//--- variables para los identificadores de gráficos
   long currChart,prevChart=ChartFirst();
   int i=0,limit=100;
   Print("ChartFirst = ",ChartSymbol(prevChart)," ID = ",prevChart);
   while(i<limit)// seguramente no tenemos más de 100 gráficos abiertos
     {
      currChart=ChartNext(prevChart); // a base del anterior obtenemos un gráfico nuev
      if(currChart<0) break;          // hemos llegado al final de la lista de gráfico
      Print(i,ChartSymbol(currChart)," ID = ",currChart);
      prevChart=currChart;// vamos a guardar el identificador del gráfico corriente pa
      i++;// no olvidemos aumentar el contador
     }
```

© 2000-2025, MetaQuotes Ltd.

ChartClose

Cierra el gráfico especificado.

```
bool  ChartClose(
   long  chart_id=0      // identificador del gráfico
   );
```

Parámetros

chart_id=0

[in] Identificador del gráfico. 0 significa el gráfico actual.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- abrirmos un nuevo gráfico con el símbolo y periodo como el gráfico actual
   long chart_id=ChartOpen(_Symbol, _Period);
   if(chart_id==0)
     {
      Print("ChartOpen() failed. Error ", GetLastError());
      return;
     }

//--- imprimimos en el registro los parámetros del gráfico abierto
   PrintFormat("A new chart of the %s symbol has been opened with a period of %s and C
               _Symbol, StringSubstr(EnumToString(_Period), 7), chart_id);

//--- esperamos dos segundos y cerramos el gráfico recién abierto
   PrintFormat("Waiting 3 seconds before closing a newly opened chart with ID %I64d ..
   Sleep(3000);
   ResetLastError();
   if(!ChartClose(chart_id))
     {
      Print("ChartClose() failed. Error ", GetLastError());
      return;
     }

//--- imprimimos un mensaje en el registro sobre el cierre exitoso del gráfico
   PrintFormat("The chart with ID %I64d was successfully closed", chart_id);
   /*
   resultado:
   A new chart of the GBPUSD symbol has been opened with a period of M1 and ChartID 13
```

© 2000-2025, MetaQuotes Ltd.

```
   Waiting 3 seconds before closing a newly opened chart with ID 133346697706632016 ..
   The chart with ID 133346697706632016 was successfully closed
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSymbol

Devuelve el nombre del símbolo del gráfico especificado.

```
string  ChartSymbol(
   long  chart_id=0      // identificador del gráfico
   );
```

Parámetros

chart_id=0

[in] Identificador del gráfico. 0 significa el gráfico actual.

Valor devuelto

Si el gráfico no existe, se devuelve una cadena vacía.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el símbolo del gráfico actual y mostramos el valor resultante en el re
   string chart_symbol = ChartSymbol();
   Print("Current chart symbol: ", chart_symbol);

//--- tomamos el identificador de un gráfico existente (en este caso, el actual)
   long chart_id=ChartID();
   chart_symbol=ChartSymbol(chart_id);
   PrintFormat("Chart symbol with ID %I64d: %s", chart_id, chart_symbol);

//--- establecemos el identificador aleatorio del gráfico al obtener el símbolo
   chart_symbol = ChartSymbol(1234567890);
   if(chart_symbol=="")
      Print("The chart with ID 1234567890 does not exist");
   else
      Print("Chart symbol with ID 1234567890: ", chart_symbol);
   /*
  resultado:
   Current chart symbol: GBPUSD
   Chart symbol with ID 132966427583395104: GBPUSD
   The chart with ID 1234567890 does not exist
   */
  }
```

Véase también

ChartSetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

ChartPeriod

Devuelve el valor del período del gráfico especificado.

```
ENUM_TIMEFRAMES  ChartPeriod(
   long  chart_id=0      // identificador del gráfico
   );
```

Parámetros

chart_id=0

[in] Identificador del gráfico. 0 significa el gráfico actual.

Valor devuelto

Valor del tipo ENUM_TIMEFRAMES. Si el gráfico no existe, se devuelve 0.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el periodo del gráfico actual y mostramos el valor resultante en el re
   ENUM_TIMEFRAMES period=ChartPeriod();
   Print("Current chart period: ", EnumToString(period));

//--- tomamos el identificador de un gráfico existente (en este caso, el actual)
   long chart_id=ChartID();
   period=ChartPeriod(chart_id);
   PrintFormat("Chart period with ID %I64d: %s", chart_id, EnumToString(period));

//--- establecemos un identificador de gráfico aleatorio
   period=ChartPeriod(1234567890);
   if(period==0)
      Print("The chart with ID 1234567890 does not exist");
   else
      Print("Chart period with ID 1234567890: ", EnumToString(period));
   /*
   resultado:
   Current chart period: PERIOD_M15
   Chart period with ID 133510090240498505: PERIOD_M15
   The chart with ID 1234567890 does not exist
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartRedraw

Activa el redibujo forzado de un gráfico especificado.

```
void  ChartRedraw(
   long  chart_id=0      // identificador del gráfico
   );
```

Parámetros

chart_id=0

[in] Identificador del gráfico. 0 significa el gráfico actual.

Nota

Suele usarse después del cambio de las propiedades de objetos.

Ejemplo:

```
#define   WIDTH      50    // anchura del rectángulo en barras

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el identificador del gráfico actual y establecemos el nombre del objet
   long   chart_id = ChartID();
   string obj_name = MQLInfoString(MQL_PROGRAM_NAME)+"_RectLabel";

//--- obtenemos las dos coordenadas de tiempo del rectángulo
   datetime time1  = iTime(_Symbol, _Period, WIDTH);
   datetime time2  = iTime(_Symbol, _Period, 0);
   if(time1==0 || time2==0)
     {
      Print("Error getting time ", GetLastError());
      return;
     }

//--- obtenemos los precios máximo y mínimo en el rango de anchura del rectángulo
   double price1 = HighestHigh(_Symbol, _Period, 0, WIDTH);
   double price2 = LowestLow(_Symbol, _Period, 0, WIDTH);
   if(price1==EMPTY_VALUE || price2==EMPTY_VALUE)
      return;

//--- creamos un objeto de rectángulo
   Print("Create a wheat-colored rectangle");
   if(!ObjectCreate(chart_id, obj_name, OBJ_RECTANGLE, 0, time1, price1, time2, price2
     {
      Print("ObjectCreate() failed. Error ", GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- rellenamos el rectángulo con el color inicial
   ObjectSetInteger(chart_id, obj_name, OBJPROP_FILL, true);
   ObjectSetInteger(chart_id, obj_name, OBJPROP_BACK, true);
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrWheat);
   ChartRedraw();

//--- esperamos un segundo, rellenamos el rectángulo con el color "DodgerBlue" y actua
   Sleep(1000);
   Print("Change color to DodgerBlue");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrDodgerBlue);
   ChartRedraw();

//--- esperamos un segundo, rellenamos el rectángulo con el color "DodgerBlue" y actua
   Sleep(1000);
   Print("Change color to LimeGreen");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrLimeGreen);
   ChartRedraw();

//--- esperamos un segundo, rellenamos el rectángulo con el color "OrangeRed" y actual
   Sleep(1000);
   Print("Change color to OrangeRed");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrOrangeRed);
   ChartRedraw();

//--- esperamos un segundo, rellenamos el rectángulo con el color "Wheat" y actualizar
   Sleep(1000);
   Print("Reset color to original");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrWheat);
   ChartRedraw();

//--- transcurridos tres segundos, eliminamos el objeto del gráfico
   Sleep(3000);
   Print("Delete the rectangle");
   ObjectDelete(chart_id, obj_name);
  }
//+------------------------------------------------------------------+
//| Retorna el High máximo en el intervalo de barras indicado        |
//+------------------------------------------------------------------+
double HighestHigh(const string symbol, const ENUM_TIMEFRAMES timeframe, const uint st
  {
   ResetLastError();
   int index=iHighest(symbol, timeframe, MODE_HIGH, count, start);
   if(index==-1)
     {
      PrintFormat("%s: iHighest() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   GetLastError();
   double price=iHigh(symbol, timeframe, index);
   if(price==0)
     {
      PrintFormat("%s: iHigh() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   return(price);
  }
//+------------------------------------------------------------------+
//| Retorna el Low mínimo en el intervalo de barras indicado         |
//+------------------------------------------------------------------+
double LowestLow(const string symbol, const ENUM_TIMEFRAMES timeframe, const uint sta
  {
   ResetLastError();
   int index=iLowest(symbol, timeframe, MODE_LOW, count, start);
   if(index==-1)
     {
      PrintFormat("%s: iLowest() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   GetLastError();
   double price=iLow(symbol, timeframe, index);
   if(price==0)
     {
      PrintFormat("%s: iLow() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   return(price);
  }
```

Véase también

Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

ChartSetDouble

Establece el valor para la propiedad correspondiente del gráfico especificado. La propiedad del gráfico debe ser del tipo double. El comando se añade a la cola de los mensajes del gráfico y se ejecuta sólo después del procesamiento de todos los comandos anteriores.

```
bool  ChartSetDouble(
   long                          chart_id,      // identificador del gráfico
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,       // identificador de la propiedad
   double                        value          // valor
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

prop_id

[in] Identificador de la propiedad del gráfico. Su valor puede ser uno de los valores de la enumeración ENUM_CHART_PROPERTY_DOUBLE (excepto las propiedades read-only).

value

[in] Valor de la propiedad.

Valor devuelto

Si el comando se coloca con éxito a la cola del gráfico, devuelve true, de lo contrario, devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

La función es asincrónica, esto significa que la función no espera la ejecución de un comando colocado con éxito en la cola del gráfico especificado, sino que devuelve inmediatamente el control. La propiedad cambiará solo después de que el comando haya sido procesado en la cola del gráfico. Para que el comando se ejecute de inmediato en la cola del gráfico, hay que llamar a la función ChartRedraw.

Si resulta necesario cambiar de inmediato varias propiedades del gráfico, entonces deberemos ejecutar las funciones correspondientes (ChartSetString, ChartSetDouble, ChartSetString) en un bloque de código y después llamar una vez ChartRedraw.

Para comprobar el resultado de la ejecución, podemos usar la función que solicita la propiedad del gráfico indicada (ChartGetInteger, ChartGetDouble, ChartSetString). En este caso, además, hay que tener en cuenta que estas funciones son sincrónicas y esperan el resultado de la ejecución.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el indentificador del gráfico actual
   long chart_id = ChartID();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- recordamos la separación inicial de la barra cero desde el borde derecho en porc
   double shift = ChartGetDouble(chart_id, CHART_SHIFT_SIZE);
   bool   scroll= ChartGetInteger(chart_id,CHART_AUTOSCROLL);

//--- restablecemos la bandera de desplazamiento automático del gráfico
   ChartSetInteger(chart_id, CHART_AUTOSCROLL, true);
   PrintFormat("Initial chart shift size: %.2f", shift);

//--- en un cilo de 2.0% a 50.0% con un paso en 0.5%
   for(int i=20; i<=500; i+=5)
     {
      //--- establecemos el tamaño de la separación en i / 10 e imprimimos el valor es
      ChartSetDouble(chart_id, CHART_SHIFT_SIZE, i/10.0);
      PrintFormat("Set chart shift size to %.1f%%", i/10.0);

      //--- mantenemos la barra cero del gráfico a la distancia de separación estable
      ChartNavigate(chart_id, CHART_END, 0);

      //--- esperamos un poco
      Sleep(16);
     }

//--- establecemos la separación inicial y el desplazamiento automático del gráfico
   Print("Set the chart shift size to the initial value of ", shift);
   ChartSetDouble(ChartID(), CHART_SHIFT_SIZE, shift);
   ChartSetInteger(chart_id, CHART_AUTOSCROLL, scroll);
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSetInteger

Establece el valor para la propiedad correspondiente del gráfico especificado. La propiedad del gráfico debe ser del tipo datetime, int, color, bool o char. El comando se añade a la cola de los mensajes del gráfico y se ejecuta sólo después del procesamiento de todos los comandos anteriores.

```
bool  ChartSetInteger(
   long                          chart_id,      // identificador del gráfico
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // identificador de la propiedad
   long                          value          // valor
   );
```

Establece un valor para la propiedad correspondiente de la subventana indicada.

```
bool  ChartSetInteger(
   long                          chart_id,      // identificador del gráfico
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // identificador de la propiedad
   int                           sub_window,    // número de subventana
   long                          value          // valor
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

prop_id

[in] Identificador de la propiedad del gráfico. Su valor puede ser uno de los valores de la enumeración ENUM_CHART_PROPERTY_INTEGER (excepto las propiedades read-only).

sub_window

[in] Número de subventana del gráfico. Para la primera variante por defecto el valor es igual a 0 (ventana principal del gráfico). La mayoría de las propiedades no requiere indicar el número de subventana.

value

[in] Valor de la propiedad.

Valor devuelto

Si el comando se coloca con éxito a la cola del gráfico, devuelve true, de lo contrario, devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

La función es asincrónica, esto significa que la función no espera la ejecución de un comando colocado con éxito en la cola del gráfico especificado, sino que devuelve inmediatamente el control. La propiedad cambiará solo después de que el comando haya sido procesado en la cola del gráfico. Para que el comando se ejecute de inmediato en la cola del gráfico, hay que llamar a la función ChartRedraw.

© 2000-2025, MetaQuotes Ltd.

Si resulta necesario cambiar de inmediato varias propiedades del gráfico, entonces deberemos ejecutar las funciones correspondientes (ChartSetString, ChartSetDouble, ChartSetString) en un bloque de código y después llamar una vez ChartRedraw.

Para comprobar el resultado de la ejecución, podemos usar la función que solicita la propiedad del gráfico indicada (ChartGetInteger, ChartGetDouble, ChartSetString). En este caso, además, hay que tener en cuenta que estas funciones son sincrónicas y esperan el resultado de la ejecución.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- habilitando mensajes sobre el desplazamiento del ratón por la ventana del gráfic
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- la actualización forzosa de las propiedades del gráfico garantiza la preparación
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse left
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse right
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse middle
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse first X key
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse second X key
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift key
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control key
   return(res);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSetString

Establece el valor para la propiedad correspondiente del gráfico especificado. La propiedad del gráfico debe ser string. El comando se añade a la cola de los mensajes del gráfico y se ejecuta sólo después del procesamiento de todos los comandos anteriores.

```
bool  ChartSetString(
   long                          chart_id,      // identificador del gráfico
   ENUM_CHART_PROPERTY_STRING    prop_id,       // identificador de la propiedad
   string                        str_value      // valor
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

prop_id

[in] Identificador de la propiedad del gráfico. Su valor puede ser uno de los valores de la enumeración ENUM_CHART_PROPERTY_STRING (salvo read-only).

str_value

[in] Cadena para establecer la propiedad. La longitud de la cadena no puede superar 2045 símbolos (los que sobran, serán recortados).

Valor devuelto

Si el comando se coloca con éxito a la cola del gráfico, devuelve true, de lo contrario, devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

La función ChartSetString puede ser usada para visualizar los comentarios en el gráfico en vez de la función Comment.

La función es asincrónica, esto significa que la función no espera la ejecución de un comando colocado con éxito en la cola del gráfico especificado, sino que devuelve inmediatamente el control. La propiedad cambiará solo después de que el comando haya sido procesado en la cola del gráfico. Para que el comando se ejecute de inmediato en la cola del gráfico, hay que llamar a la función ChartRedraw.

Si resulta necesario cambiar de inmediato varias propiedades del gráfico, entonces deberemos ejecutar las funciones correspondientes (ChartSetString, ChartSetDouble, ChartSetString) en un bloque de código y después llamar una vez ChartRedraw.

Para comprobar el resultado de la ejecución, podemos usar la función que solicita la propiedad del gráfico indicada (ChartGetInteger, ChartGetDouble, ChartSetString). En este caso, además, hay que tener en cuenta que estas funciones son sincrónicas y esperan el resultado de la ejecución.

Ejemplo:

```
void OnTick()
  {
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
   string comment=StringFormat("Выводим цены:\nAsk = %G\nBid = %G\nSpread = %d",
                               Ask,Bid,Spread);
   ChartSetString(0,CHART_COMMENT,comment);
  }
```

Véase también

Comment, ChartGetString

© 2000-2025, MetaQuotes Ltd.

ChartGetDouble

Devuelve el valor para la propiedad correspondiente del gráfico especificado. La propiedad del gráfico debe ser del tipo double. Existen 2 variantes de la función.

1 Inmediatamente devuelve el valor de la propiedad.

```
double  ChartGetDouble(
   long                          chart_id,         // identificador del gráfico
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,          // identificador de la propiedad
   int                           sub_window=0      // número de subventana, si hace falt
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  ChartGetDouble(
   long                          chart_id,         // identificador del gráfico
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,          // identificador de la propiedad
   int                           sub_window,       // número de subventana
   double&                       double_var        // aquí recibimos el valor de la prop
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

prop_id

[in] Identificador de la propiedad del gráfico. Su valor puede ser uno de los valores de la enumeración ENUM_CHART_PROPERTY_DOUBLE.

sub_window

[in] Número de subventana del gráfico. Para la primera variante por defecto el valor es igual a 0 (ventana principal del gráfico). La mayoría de las propiedades no requiere indicar el número de subventana.

double_var

[out] Variable del tipo double que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo double.

En segundo caso devuelve true, si dicha propiedad está disponible y su valor ha sido pasado en la variable double_var, de lo contrario devuelve false. Para obtener más detalles sobre el error hay que llamar a la función GetLastError().

Nota

La función es sincrónica, esto significa que esperará a que se ejecuten todos los comandos ubicados en la cola del gráfico antes de ser llamada.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,0);
   double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,0);
   Print("CHART_PRICE_MIN = ",priceMin);
   Print("CHART_PRICE_MAX = ",priceMax);
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartGetInteger

Devuelve el valor para la propiedad correspondiente del gráfico especificado. La propiedad del gráfico debe ser del tipo datetime, int o bool. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
long  ChartGetInteger(
   long                          chart_id,         // identificador del gráfico
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // identificador de la propiedad
   int                           sub_window=0      // número de subventana, si hace fal
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  ChartGetInteger(
   long                          chart_id,         // identificador del gráfico
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // identificador de la propiedad
   int                           sub_window,       // número de subventana
   long&                         long_var          // aquí recibimos el valor de la pro
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

prop_id

[in] Identificador de la propiedad del gráfico. Su valor puede ser uno de los valores de la enumeración ENUM_CHART_PROPERTY_INTEGER.

sub_window

[in] Número de subventana del gráfico. Para la primera variante por defecto el valor es igual a 0 (ventana principal del gráfico). La mayoría de las propiedades no requiere indicar el número de subventana.

long_var

[out] Variable del tipo long que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo long.

En segundo caso devuelve true, si dicha propiedad estça disponible y su valor ha sido pasado en la variable long_var, de lo contrario devuelve false. Para obtener más detalles sobre el error,hay que llamar a la función GetLastError().

Nota

La función es sincrónica, esto significa que esperará a que se ejecuten todos los comandos ubicados en la cola del gráfico antes de ser llamada.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   Print("CHART_HEIGHT_IN_PIXELS = ",height," pixels");
   Print("CHART_WIDTH_IN_PIXELS = ",width," pixels");
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartGetString

Devuelve el valor para la propiedad correspondiente del gráfico especificado. La propiedad del gráfico debe ser del tipo string. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
string  ChartGetString(
   long                          chart_id,         // identificador del gráfico
   ENUM_CHART_PROPERTY_STRING    prop_id           // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  ChartGetString(
   long                          chart_id,         // identificador del gráfico
   ENUM_CHART_PROPERTY_STRING    prop_id,          // identificador de la propiedad
   string&                       string_var        // aquí recibimos el valor de la prop
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

prop_id

[in] Identificador de la propiedad del gráfico. Su valor puede ser uno de los valores de la enumeración ENUM_CHART_PROPERTY_STRING.

string_var

[out] Variable del tipo string que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo string.

En segundo caso devuelve true, si dicha propiedad estça disponible y su valor ha sido pasado en la variable string_var, de lo contrario devuelve false. Para obtener más detalles sobre el error hay que llamar a la función GetLastError().

Nota

La función ChartGetString puede ser usada para leer los comentarios se visualizan en el gráfico usando las funciones Comment o ChartSetString.

La función es sincrónica, esto significa que esperará a que se ejecuten todos los comandos ubicados en la cola del gráfico antes de ser llamada.

Ejemplo:

```
void OnStart()
  {
   ChartSetString(0,CHART_COMMENT,"Test comment.\nSecond line.\nThird!");
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartRedraw();
   Sleep(1000);
   string comm=ChartGetString(0,CHART_COMMENT);
   Print(comm);
  }
```

Véase también

Comment, ChartSetString

© 2000-2025, MetaQuotes Ltd.

ChartNavigate

Desplaza el gráfico especificado a una cantidad de barras especificada respecto a la posición del gráfico indicada.

```
bool  ChartNavigate(
   long                  chart_id,      // identificador del gráfico
   ENUM_CHART_POSITION   position,      // posición
   int                   shift=0        // valor del desplazamiento
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

position

[in] Posición del gráfico respecto a la que va a realizarse el desplazamiento. Su valor puede ser uno de los valores de la enumeración ENUM_CHART_POSITION.

shift=0

[in] Número de barras al que hay que mover el gráfico. El valor positivo supone el desplazamiento a la derecha (al final del gráfico), el valor negativo significa el desplazamiento a la izquierda (al principio del gráfico). El desplazamiento cero tiene sentido cuando navegamos al principio o al final del gráfico.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el manejador (handle) del gráfico actual
   long handle=ChartID();
   string comm="";
   if(handle>0) // si ha salido bien, realizaremos ajustes adicionales
     {
      //--- desactivaremos el desplazamiento automático
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- establecemos la sangría del borde derecho del gráfico
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- mostramos en forma de velas
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- establecemos el modo de visualización de los volúmenes de ticks
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);

      //--- preparamos el texto para mostrar en Comment()
```

© 2000-2025, MetaQuotes Ltd.

```
   comm="Desplazamos a 10 barras a la derecha desde el inicio del historial";
   //--- mostramos el comentario
   Comment(comm);
   //--- desplazamos a 10 barras a la derecha desde el inicio del historial
   ChartNavigate(handle,CHART_BEGIN,10);
   //--- obtenemos el número de la primera barra visible en el gráfico (numeración
   long first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   //--- añadimos el carácter de nueva línea
   comm=comm+"\r\n";
   //--- añadimos comentario
   comm=comm+"La primera barra en el gráfico tiene el número "+IntegerToString(firs
   //--- mostramos el comentario
   Comment(comm);
   //--- esperaremos 5 segundos para que nos de tiempo a ver cómo se desplaza el gr
   Sleep(5000);

   //--- completaremos el texto del comentario
   comm=comm+"\r\n"+"Desplazamos a 10 barras a la izquierda desde el borde derecho
   Comment(comm);
   //--- desplazamos a 10 barras a la izquierda desde el borde derecho del gráfico
   ChartNavigate(handle,CHART_END,-10);
   //--- obtenemos el número de la primera barra visible en el gráfico (numeración
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"La primera barra en el gráfico tiene el número "+IntegerToString(firs
   Comment(comm);
   //--- esperaremos 5 segundos para que nos de tiempo a ver cómo se desplaza el gr
   Sleep(5000);

   //--- nuevo bloque del desplazamiento del gráfico
   comm=comm+"\r\n"+"Desplazamos a 300 barras a la derecha desde el inicio del hist
   Comment(comm);
   //--- desplazamos a 300 barras a la derecha desde el inicio del historial
   ChartNavigate(handle,CHART_BEGIN,300);
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"La primera barra en el gráfico tiene el número "+IntegerToString(firs
   Comment(comm);
   //--- esperaremos 5 segundos para que nos de tiempo a ver cómo se desplaza el gr
   Sleep(5000);

   //--- nuevo bloque del desplazamiento del gráfico
   comm=comm+"\r\n"+"Desplazamos a 300 barras a la izquierda desde el borde derecho
   Comment(comm);
   //--- desplazamos a 300 barras a la izquierda desde el borde derecho del gráfico
   ChartNavigate(handle,CHART_END,-300);
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"La primera barra en el gráfico tiene el número "+IntegerToString(firs
```

© 2000-2025, MetaQuotes Ltd.

```
      Comment(comm);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartID

Devuelve el identificador del gráfico corriente.

```
long  ChartID();
```

Valor devuelto

Valor del tipo long.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- variables para identificadores de los gráficos
   long curr_chart=ChartFirst();
   int i=0;
   //--- imprimimos en el registro los datos del primer gráfico
   PrintFormat("Chart[%d] ID: %I64d,  symbol: %s", i, curr_chart, ChartSymbol(curr_cha

//--- por ahora no hemos llegado al límite de gráficos abiertos (CHARTS_MAX)
   while(!IsStopped() && i < CHARTS_MAX)
     {
      //--- aumentamos el contador de gráficos
      i++;
      //--- usando como base el gráfico anterior, obtenemos el identificador del sigui
      curr_chart=ChartNext(curr_chart);

      //--- si hemos alcanzado el final de la lista de gráficos, finalizamos el ciclo
      if(curr_chart<0)
         break;

      //--- imprimimos en el registro los datos del siguiente gráfico
      PrintFormat("Chart[%d] ID: %I64d,  symbol: %s", i, curr_chart, ChartSymbol(curr_
     }
   /*
   resultado:
   Chart[0] ID: 133246248352168440,  symbol: EURUSD
   Chart[1] ID: 133346697706632015,  symbol: USDJPY
   Chart[2] ID: 133246248352168439,  symbol: GBPUSD
   Chart[3] ID: 133346697706632009,  symbol: RU000A103661
   Chart[4] ID: 133346697706632010,  symbol: AEM4
   Chart[5] ID: 133346697706632011,  symbol: AA.SPB
   Chart[6] ID: 133346697706632012,  symbol: ALLFUTMIX
   Chart[7] ID: 133346697706632013,  symbol: EURUSD
   Chart[8] ID: 133346697706632014,  symbol: SBER
   */
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorAdd

Añade un indicador con el manejador (handle) especificado a la ventana del gráfico especificada. El indicador y el gráfico tienen que generarse basándose en el mismo símbolo y en el mismo período de tiempo.

```
bool  ChartIndicatorAdd(
   long  chart_id,                  // identificador del gráfico
   int   sub_window                 // número de subventana
   int   indicator_handle           // manejador del indicador
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

sub_window

[in] Número de subventana del gráfico. 0 significa la ventana principal del gráfico. Para añadir un indicador a una ventana nueva, el parámetro debe ser un uno más que el índice de la última ventana existente, es decir debe ser igual a CHART_WINDOWS_TOTAL. Si el valor del parámetro sobrepasa el valor de CHART_WINDOWS_TOTAL, entonces la nueva ventana no se creará y el indicador no será agregado.

indicator_handle

[in] Manejador del indicador.

Valor devuelto

Devuelve true en caso del éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError(). El error 4114 significa que el gráfico y el indicador que se añade se diferencian por el símbolo o por el período de tiempo.

Nota

Si a la ventana principal del gráfico se le añade un indicador que tiene que ser dibujado en una ventana separada (por ejemplo, indicador built-in iMACD o un indicador personalizado con la propiedad especificada #property indicator_separate_window), entonces puede ocurrir que este indicador se quede invisible, aunque va a figurar en la lista de indicadores. Eso quiere decir que la escala de este indicador se diferencia de la escala del gráfico de precios, y los valores del indicador insertado no entran en la extensión visualizada del gráfico de precios. En este caso GetLastError() va a devolver el código nulo que indica en la ausencia de errores. Se puede observar el valor de este indicador "invisible" en la "Ventana de Datos", así como obtenerlo desde otros programas MQL5.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#property description "El EA para demostrar el trabajo con la función ChartIndicatorAd
#property description "Después de arrancarlo en el gráfico (y recibir el error en el I
#property description "las propiedades del EA y establezca los parámetros correctos de
#property description "El indicador MACD sera agregado al gráfico."

//--- input parameters
input string          symbol="AUDUSD";     // nombre del símbolo
input ENUM_TIMEFRAMES period=PERIOD_M12;   // período de tiempo
input int    fast_ema_period=12;           // período de media rápida MACD
input int    slow_ema_period=26;           // período de media lenta MACD
input int      signal_period=9;            // período del promedio de diferencia
input ENUM_APPLIED_PRICE apr=PRICE_CLOSE; // tipo del precio para el cálculo MACD

int indicator_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_period,
//--- intentamos insertar el indicador en el gráfico
   if(!AddIndicator())
     {
      //--- la función AddIndicator() ha negado a insertar el indicador en el gráfico
      int answer=MessageBox("¿Desea intentar agregar MACD al gráfico de cualquier modo
                            "El símbolo y/o el período de tiempo no han sido estableci
                            MB_YESNO // se mostrarán los botones de selección "Yes" y
                            );
      //--- si el usuario aun así insiste en el uso incorrecto de ChartIndicatorAdd()
      if(answer==IDYES)
        {
         //--- primero vamos a reflejarlo en el Diario
         PrintFormat("¡Atención! %s: Intentemos agregar el indicador MACD(%s/%s) al gr
                     __FUNCTION__,symbol,EnumToString(period),_Symbol,EnumToString(_Pe
         //--- obtenemos el número de una nueva subventana en la que tratamos de inser
         int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
         //--- ahora hagamos el intento condenado al error
         if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
            PrintFormat("Fallo al agergar el indicador MACD a la ventana %d del gráfic
                        subwindow,GetLastError());
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
// el EA no hace nada
  }
//+------------------------------------------------------------------+
//| La función para comprobar y añadir el indicador al gráfico       |
//+------------------------------------------------------------------+
bool AddIndicator()
  {
//--- mostramos el mensaje
   string message;
//--- comprobamos si el símbolo del indicador coincide con el símbolo del gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
   if(symbol!=_Symbol)
     {
      message="Demostración del usu de la función Demo_ChartIndicatorAdd():";
      message=message+"\r\n";
      message=message+"No se puede agregar al gráfico un indicador calculado para otro
      message=message+"\r\n";
      message=message+"Especifique el símbolo del gráfico en las propiedades del EA -
      Alert(message);
      //--- salida anticipada, no vamos a añadir el indicador al gráfico
      return false;
     }
//--- comprobaremos si el periodo del indicador coincide con el período del gráfico
   if(period!=_Period)
     {
      message="No se puede agregar al gráfico un indicador calculado para otro períod
      message=message+"\r\n";
      message=message+"Especifique el período del gráfico en las propiedades del EA -
      Alert(message);
      //--- salida anticipada, no vamos a añadir el indicador al gráfico
      return false;
     }
//--- todas las pruebas pasadas, el símbolo y el período del indicador corresponden a
   if(indicator_handle==INVALID_HANDLE)
     {
      Print(__FUNCTION__,"  Creamos el indicador MACD");
      indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_peri
      if(indicator_handle==INVALID_HANDLE)
        {
         Print("Generación del indicador MACD fallida. Código del error ",GetLastError
        }
     }
//--- reseteamos el código del error
   ResetLastError();
//--- agregamos el indicador al gráfico
   Print(__FUNCTION__,"  Agregamos el indicador MACD al gráfico");
   Print("MACD construida sobre ",symbol,"/",EnumToString(period));
//--- obtenemos el número de nueva subventana a la que vamos a agregar el indicador MA
   int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   PrintFormat("Agregamos el indicador MACD a la ventana %d del gráfico",subwindow);
   if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
     {
      PrintFormat("Fallo al agergar el indicador MACD a la ventana %d del gráfico. Cód
                  subwindow,GetLastError());
     }
//--- el indicador ha sido insertado al gráfico con éxito
   return(true);
  }
```

Véase también

ChartIndicatorDelete(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorDelete

Quita el indicador con el nombre especificado de la ventana del gráfico especificada.

```
bool  ChartIndicatorDelete(
   long           chart_id,              // identificador del gráfico
   int            sub_window             // número de subventana
   const string   indicator_shortname    // nombre breve del indicador
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

sub_window

[in] Número de subventana del gráfico. 0 significa la ventana principal del gráfico.

const indicator_shortname

[in] Nombre breve del indicador que se establece en la propiedad INDICATOR_SHORTNAME por la función IndicatorSetString(). Para obtener el nombre breve del indicador se usa la función ChartIndicatorName().

Valor devuelto

La función devuelve true si el indicador se elimina con éxito, de lo contrario se devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Si en la subventana del gráfico especificada hay varios indicadores con el mismo nombre breve, se elimina el que va primero.

Si en el mismo gráfico están construidos otros indicadores a base de los valores del indicador que se elimina, éstos también serán eliminados.

No se debe confundir el nombre breve del indicador con el nombre del archivo que se indica durante la creación del indicador por las funciones iCustom() y IndicatorCreate(). Si el nombre breve del indicador no está establecido de una manera explícita, entonces durante la compilación ahí se indica el nombre del archivo que contiene el código fuente del indicador.

Si un indicador se quita del gráfico, esto no significa que su parte de cálculo también será eliminada de la memoria del terminal. Para liberar el manejador del indicador, hay que usar la funciónIndicatorRelease().

Se debe formar correctamente el nombre breve del indicador. Este nombre se escribe en la propiedad INDICATOR_SHORTNAME mediante la función IndicatorSetString(). Es recomendable que el nombre breve contenga los valores de los parámetros de entrada del indicador, puesto que en la función ChartIndicatorDelete() la identificación del indicador que se quita del gráfico se realiza precisamente por el nombre breve.

Ejemplo de eliminación de un indicador tras el fallo de inicialización:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                  Demo_ChartIndicatorDelete.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      first_param=1;
input int      second_param=2;
input int      third_param=3;
input bool     wrong_init=true;
//--- indicator buffers
double         HistogramBuffer[];
string         shortname;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int res=INIT_SUCCEEDED;
//--- vinculamos el array HistogramBuffer al búfer de indicador
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- generamos el nombre breve del indicador a base de los parámetros de entrada
   shortname=StringFormat("Demo_ChartIndicatorDelete(%d,%d,%d)",
                          first_param,second_param,third_param);
   IndicatorSetString(INDICATOR_SHORTNAME,shortname);
//--- si está establecida la finalización forzosa del indicador, devolvemos el valor r
   if(wrong_init) res=INIT_FAILED;
   return(res);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- posición inicial para trabajar en el ciclo
   int start=prev_calculated-1;
   if(start<0) start=0;
//--- llenamos el búfer de indicador con los valores
   for(int i=start;i<rates_total;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      HistogramBuffer[i]=close[i];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| manejador de eventos Deinit                                      |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   PrintFormat("%s: Código de la causa de deinicialización=%d",__FUNCTION__,reason);
   if(reason==REASON_INITFAILED)
     {
      PrintFormat("El indicador con el nombre breve %s (archivo %s) elimina a sí mismo
      int window=ChartWindowFind();
      bool res=ChartIndicatorDelete(0,window,shortname);
      //--- analizaremos el resultado de la llamada a ChartIndicatorDelete()
      if(!res)
        {
         PrintFormat("Fallo al eliminar el indicador %s desde la ventana #%d. Código d
                     shortname,window,GetLastError());
        }
     }
  }
```

Véase también

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorGet

Devuelve el manejador del indicador con el nombre corto especificado en la ventana del gráfico especificada.

```
int  ChartIndicatorGet(
   long           chart_id,              // identificador del gráfico
   int            sub_window             // número de subventana
   const string   indicator_shortname    // nombre corto del indicador
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

sub_window

[in] Número de subventana del gráfico. 0 significa la ventana principal del gráfico.

const indicator_shortname

[in] El nombre corto del indicador que se establece en la propiedad INDICATOR_SHORTNAME a través de la función IndicatorSetString(). Para obtener el nombre corto del indicador, utilice la función ChartIndicatorName().

Valor devuelto

Devuelve el manejador del indicador en caso de la ejecución con éxito, de lo contrario INVALID_HANDLE. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

El manejador de indicador obtenido con la ayuda de la función ChartIndicatorGet() aumenta el contador interno de uso de este indicador. El sistema de ejecución del terminal mantiene cargados todos los indicadores cuyo contador es superior a cero. Por eso, es necesario liberar de forma instantánea y explícita cualquier manejador de indicador innecesario, con la ayuda de IndicatorRelease() y en el mismo programa que lo ha recibido, como se muestra en el ejemplo más abajo. En caso contrario, no será posible encontrar el manejador "abandonado" y liberarlo desde otro programa.

A la hora de crear un indicador, hace falta formar de forma correcta su nombre corto que a través de la función IndicatorSetString() se escribe en la propiedad INDICATOR_SHORTNAME. Se recomienda que el nombre corto contenga los valores de los parámetros de entrada del indicador, ya que la identificación del indicador en la función ChartIndicatorGet() se realiza precisamente por el nombre corto.

Otro modo de identificar un indicador es recibir la lista de sus parámetros para el manejador establecido a través de la función IndicatorParameters() y luego realizar el análisis de valores recibidos.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- número de ventanas en el gráfico (siempre hay por lo menos una ventana princi
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   //--- repasamos las ventanas
   for(int w=0;w<windows;w++)
     {
      //--- cuántos indicadores hay en esta ventana/subventana
      int total=ChartIndicatorsTotal(0,w);
      //--- repasamos todos los indicadores en la ventana
      for(int i=0;i<total;i++)
        {
         //--- obtenemos el nombre corto del indicador
         string name=ChartIndicatorName(0,w,i);
         //--- obtenemos el manejador del indicador
         int handle=ChartIndicatorGet(0,w,name);
         //--- mostramos en el diario
         PrintFormat("Window=%d,  index=%d,  name=%s,  handle=%d",w,i,name,handle);
         //--- release handle
         IndicatorRelease(handle);
        }
     }
  }
```

Véase también

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), IndicatorParameters()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorName

Devuelve el nombre breve del indicador según su número en la lista de indicadores en la determinada ventana del gráfico.

```
string  ChartIndicatorName(
   long  chart_id,      // identificador del gráfico
   int   sub_window     // número de subventana
   int   index          // índice del indicador en la lista de indicadores agregados a
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

sub_window

[in] Número de subventana del gráfico. 0 significa la ventana principal del gráfico.

index

[in] Índice del indicador en la lista de indicadores. La numeración de indicadores se empieza desde cero, es decir el primer indicador de la lista tiene el índice cero. El número de indicadores en la lista se obtiene a través de la función ChartIndicatorsTotal().

Valor devuelto

La función devuelve el nombre breve del indicador que se establece en la propiedad INDICATOR_SHORTNAME por la función IndicatorSetString(). Para obtener el nombre breve, se puede usar la función ChartIndicatorName(). Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

No se debe confundir el nombre breve del indicador con el nombre del archivo que se indica durante la creación del indicador por las funciones iCustom() y IndicatorCreate(). Si el nombre breve del indicador no está establecido de una manera explícita, entonces durante la compilación ahí se indica el nombre del archivo que contiene el código fuente del indicador.

Si un indicador se quita del gráfico, esto no significa que su parte de cálculo también será eliminada de la memoria del terminal. Para liberar el manejador del indicador, hay que usar la funciónIndicatorRelease().

Se debe formar correctamente el nombre breve del indicador. Este nombre se escribe en la propiedad INDICATOR_SHORTNAME mediante la función IndicatorSetString(). Es recomendable que el nombre breve contenga los valores de los parámetros de entrada del indicador, puesto que en la función ChartIndicatorDelete() la identificación del indicador que se quita del gráfico se realiza precisamente por el nombre breve.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- obtenemos el identificador del gráfico actual y el número de sus subventanas, in
   long chart_id = ChartID();
   int  wnd_total= (int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);

//--- iteramos en un ciclo por todas las ventanas del gráfico actual
   for(int w=0; w<wnd_total; w++)
     {
      //--- obtenemos el número de indicadores adjuntos a la ventana del gráfico indic
      int ind_total=ChartIndicatorsTotal(chart_id, w);

      //--- imprimimos el encabezado para la ventana de gráfico elegida
      PrintFormat("Chart window %d indicators: ", w);

      //--- obtenemos en un ciclo y escribimos en una variable todos los nombres de lo
      string ind_names="";
      for(int i=0; i<ind_total; i++)
        {
         ind_names+="  "+ChartIndicatorName(chart_id, w, i)+(i<ind_total-1 ? "\n": "")
        }
      //--- imprimimos en el registro la lista resultante de los nombres de todos los
      Print(ind_names);
     }
   /*
   resultado:
   Chart window 0 indicators:
     AMA(9,2,30)
     SAR(0.02,0.20)
     Fractals
   Chart window 1 indicators:
     RSI(14)
     AMA(9,2,30)
   Chart window 2 indicators:
     MFI(14)
   */
  }
```

Véase también

ChartIndicatorAdd(), ChartIndicatorDelete(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorsTotal

Devuelve el número de todos los indicadores vinculados con la determinada ventana del gráfico.

```
int  ChartIndicatorsTotal(
   long  chart_id,       // identificador del gráfico
   int   sub_window      // número de subventana
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

sub_window

[in] Número de subventana del gráfico. 0 significa la ventana principal del gráfico.

Valor devuelto

Número de indicadores en la ventana del gráfico especificada. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

La función sirve para organizar el repaso de todos los indicadores enlazados con este gráfico. El número de todas las ventanas del gráfico se puede obtener desde la propiedad CHART_WINDOWS_TOTAL mediante la función ChartGetInteger().

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el identificador del gráfico actual y el número de sus subventanas, ir
   long chart_id = ChartID();
   int  wnd_total= (int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);

//--- iteramos en un ciclo por todas las ventanas del gráfico actual
   for(int w=0; w<wnd_total; w++)
     {
      //--- obtenemos el número de indicadores adjuntos a la ventana del gráfico indic
      int ind_total=ChartIndicatorsTotal(chart_id, w);

      //--- imprimimos el número de indicadores adjuntos a la ventana del gráfico espe
      PrintFormat("Chart ID %I64d, subwindow %d. Attached indicators: %d", chart_id, w
     }
   /*
   resultado:
   Chart ID 133246248352168439, subwindow 0. Attached indicators: 3
   Chart ID 133246248352168439, subwindow 1. Attached indicators: 2
```

© 2000-2025, MetaQuotes Ltd.

```
   Chart ID 133246248352168439, subwindow 2. Attached indicators: 1
   */
  }
```

Véase también

ChartIndicatorAdd(), ChartIndicatorDelete(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2025, MetaQuotes Ltd.

ChartWindowOnDropped

Devuelve el número de la subventana del gráfico a la que el Asesor Experto o script ha sido arrastrado con el ratón. 0 significa la ventana principal del gráfico.

```
int  ChartWindowOnDropped();
```

Valor devuelto

Valor del tipo int.

Ejemplo:

```
   int myWindow=ChartWindowOnDropped();
   int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   Print("Script iniciado en la ventana #"+myWindow+
         ". El total de ventanas en el gráfico "+ChartSymbol()+": ",windowsTotal);
```

Véase también

ChartPriceOnDropped, ChartTimeOnDropped, ChartXOnDropped, ChartYOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartPriceOnDropped

Devuelve la coordinada de precios que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón.

```
double  ChartPriceOnDropped();
```

Valor devuelto

Valor del tipo double.

Ejemplo:

```
   double p=ChartPriceOnDropped();
   Print("ChartPriceOnDropped() = ",p);
```

Véase también

ChartXOnDropped, ChartYOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartTimeOnDropped

Devuelve la coordinada de tiempo que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón.

```
datetime  ChartTimeOnDropped();
```

Valor devuelto

Valor del tipo datetime.

Ejemplo:

```
   datetime t=ChartTimeOnDropped();
   Print("Script wasdropped on the "+t);
```

Véase también

ChartXOnDropped, ChartYOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartXOnDropped

Devuelve la coordinada del eje de X que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón.

```
int  ChartXOnDropped();
```

Valor devuelto

Valor de coordinada X.

Nota

El eje X va de izquierda a derecha.

Ejemplo:

```
   int X=ChartXOnDropped();
   int Y=ChartYOnDropped();
   Print("(X,Y) = ("+X+","+Y+")");
```

Véase también

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartYOnDropped

Devuelve la coordinada del eje de Y que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón.

```
int  ChartYOnDropped();
```

Valor devuelto

Valor de coordinada Y.

Nota

El eje Y va desde arriba hacia abajo.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos e imprimimos las coordenadas X e Y del punto del gráfico en el que est
   int x=ChartXOnDropped();
   int y=ChartYOnDropped();
   PrintFormat("Script dropped to coordinates X = %d, Y = %d", x, y);
   /*
   resultado:
   Script dropped to coordinates X = 429, Y = 114
   */
  }
```

Véase también

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartSetSymbolPeriod

Cambia el valor del símbolo y período del gráfico especificado. Esta función es asíncrona, es decir, envía un comando y no espera la finalización de su ejecución. El comando se añade a la cola de los mensajes del gráfico y se ejecuta sólo después del procesamiento de todos los comandos anteriores.

```
bool  ChartSetSymbolPeriod(
   long             chart_id,      // identificador del gráfico
   string           symbol,        // nombre del símbolo
   ENUM_TIMEFRAMES  period         // período
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

symbol

[in] Símbolo del gráfico. NULL significa el símbolo del gráfico corriente al que está adjuntado Asesor Experto.

period

[in] Período del gráfico (período de tiempo). Puede adquirir uno de los valores de la enumeración ENUM_TIMEFRAMES. 0 significa el período del gráfico corriente.

Valor devuelto

Si el comando se coloca con éxito a la cola del gráfico, devuelve true, de lo contrario, devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

El cambio del símbolo/período provoca la reinicialización del Asesor Experto adjuntado al gráfico correspondiente.

La llamada de ChartSetSymbolPeriod con el mismo símbolo y marco temporal se puede utilizar para actualizar el gráfico (de forma análoga al comando Refresh en el terminal). La acutalización del gráfico, a su vez, inicia el recálculo de los indicadores ligados a él. De esta forma, usted podrá calcular el indicador en el gráfico incluso en ausencia de ticks (por ejemplo, durante los festivos).

Ejemplo:

```
#define   SYMBOL    "GBPUSD"
#define   PERIOD    PERIOD_H1

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el identificador del gráfico actual, su símbolo y su periodo
   long            chart_id= ChartID();
   string          symbol  = Symbol();
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES period  = Period();

//--- informamos al registro sobre el cambio del símbolo y el periodo del gráfico actu
   PrintFormat("Change the %s symbol and the %s period of the chart %I64u to %s %s",
               symbol, TimeframeDescrioption(period), chart_id, SYMBOL, TimeframeDesc

//--- cambiamos el símbolo y el periodo del gráfico
   ChartSetSymbolPeriod(chart_id, SYMBOL, PERIOD);
   /*
   resultado:
   Change the EURUSD symbol and the M1 period of the chart 133246248352168440 to GBPUS
   */
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del periodo del gráfico                   |
//+------------------------------------------------------------------+
string TimeframeDescrioption(const ENUM_TIMEFRAMES timeframe)
  {
   return(StringSubstr(EnumToString(timeframe), 7));
  }
```

Véase también

ChartSymbol, ChartPeriod

© 2000-2025, MetaQuotes Ltd.

ChartScreenShot

Esta función proporciona la captura de pantalla del gráfico especificado en su estado actual en formato GIF, PNG o BMP, dependiendo de la extensión especificada.

```
bool  ChartScreenShot(
   long             chart_id,                  // identificador del gráfico
   string           filename,                  // nombre del archivo
   int              width,                     // ancho
   int              height,                    // alto
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // tipo de alineación
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

filename

[in] Nombre del archivo de screenshot. No puede ser más de 63 símbolos. El archivo se coloca en el directorio \Files.

width

[in] Ancho de screenshot en píxeles.

height

[in] Alto de screenshot en píxeles.

align_mode=ALIGN_RIGHT

[in] Modo output de un screenshot estrecho. Valor de enumeración ENUM_ALIGN_MODE. ALIGN_RIGHT significa la alineación por el margen derecho (output desde el final). ALIGN_LEFT significa la alineación por la izquierda.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

Si hace falta tomar un screenshot del gráfico desde una posición concreta, primero hay que posicionar el gráfico usando la función ChartNavigate(). Si el tamaño horizontal del screenshot es menos que la ventana del gráfico, se toma la parte derecha o izquierda de la ventana del gráfico dependiendo del parámetro align_mode.

Ejemplo:

```
#property description "El Asesor Experto muestra cómo se crea una serie de screenshots
#property description "utilizando la función ChartScreenShot(). Para que sea más cómod
#property description "se visualiza en el gráfico. Las macros determinan el alto y el

#define        WIDTH  800     // ancho de la imagen para llamar a ChartScreenShot()
#define        HEIGHT 600     // alto de la imagen para llamar a ChartScreenShot()
```

© 2000-2025, MetaQuotes Ltd.

```
//--- input parameters
input int      pictures=5;    // número de imágenes en la serie
int            mode=-1;       // -1 significa el desplazamiento hacia el lado derecho
int            bars_shift=300;// número de barras durante el desplazamiento del gráfic
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- desactivamos el desplazamiento automático del gráfico
   ChartSetInteger(0,CHART_AUTOSCROLL,false);
//--- establecemos la sangría del borde derecho del gráfico
   ChartSetInteger(0,CHART_SHIFT,true);
//--- mostramos el gráfico como una secuencia de velas japonesas
   ChartSetInteger(0,CHART_MODE,CHART_CANDLES);
//---
   Print("La preparación del EA para el trabajo está finalizada");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- mostrar el nombre de la función, hora de la llamada y el identificador del event
   Print(__FUNCTION__,TimeCurrent()," id=",id," mode=",mode);
//--- procesamiento del evento CHARTEVENT_CLICK ("Clic del ratón en el gráfico")
   if(id==CHARTEVENT_CLICK)
     {
      //--- desplazamiento inicial del borde del gráfico
      int pos=0;
      //--- modo de trabajo con el borde izquierdo del gráfico
      if(mode>0)
        {
         //--- desplazamos el gráfico hacia el borde izquierdo
         ChartNavigate(0,CHART_BEGIN,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- preparamos el texto a mostrar en el gráfico y el nombre para el arch
```

© 2000-2025, MetaQuotes Ltd.

```
            string name="ChartScreenShot"+"CHART_BEGIN"+string(pos)+".gif";
            //--- mostramos el nombre en el gráfico en forma del comentario
            Comment(name);
            //--- guardamos el screenshot del gráfico en la carpeta directorio_del_ter
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_LEFT))
               Print("Hemos guardado el screenshot ",name);
            //---
            pos+=bars_shift;
            //--- dejamos al usuario tiempo para que mire una nueva área del gráfico
            Sleep(3000);
            //--- desplazamos el gráfico a bars_shift a la derecha de su posición actu
            ChartNavigate(0,CHART_CURRENT_POS,bars_shift);
           }
         //--- cambio del modo al opuesto
         mode*=-1;
        }
      else // modo de trabajo con el borde derecho del gráfico
        {
         //--- desplazamos el gráfico hacia el borde derecho
         ChartNavigate(0,CHART_END,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- preparamos el texto a mostrar en el gráfico y el nombre para el arch
            string name="ChartScreenShot"+"CHART_END"+string(pos)+".gif";
            //--- mostramos el nombre en el gráfico en forma del comentario
            Comment(name);
            //--- guardamos el screenshot del gráfico en la carpeta directorio_del_ter
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_RIGHT))
               Print("Hemos guardado el screenshot ",name);
            //---
            pos+=bars_shift;
            //--- dejamos al usuario tiempo para que mire una nueva área del gráfico
            Sleep(3000);
            //--- desplazamos el gráfico a bars_shift a la derecha de su posición actu
            ChartNavigate(0,CHART_CURRENT_POS,-bars_shift);
           }
         //--- cambio del modo al opuesto
         mode*=-1;
        }
     }  // fin del procesamiento del evento CHARTEVENT_CLICK
//--- fin del manejador OnChartEvent()
  }
```

Véase también

ChartNavigate(), Recursos

© 2000-2025, MetaQuotes Ltd.

Funciones comerciales

Es el grupo de funciones que sirven para gestionar la actividad comercial.

Antes de empezar a estudiar las funciones comerciales de la plataforma, es necesario formar una idea clara sobre los términos principales: orden, operación (transacción) y posición:

- La orden es una disposición que da el cliente al broker para comprar o vender un instrumento financiero. Hay dos principales tipos de órdenes: una orden de mercado y una orden pendiente. Aparte de éstas, existen dos órdenes especiales: Take Profit y Stop Loss.
- La operación (transacción) es el hecho de comprar o vender un instrumento financiero. La compra (Buy) se realiza por el precio de oferta (Ask), mientras que la venta (Sell) — por el precio de demanda (Bid). Una operación puede ser realizada como resultado de ejecución de una orden de mercado o accionamiento de una orden pendiente. Hay que tener en cuenta que en algunas ocasiones como resultado de ejecución de una orden pueden ser varias operaciones.
- La posición es una obligación de mercado, número de contratos comprados o vendidos de un instrumento financiero. Posición larga (Long) — se trata de un instrumento financiero comprado a la espera de la subida de su precio. Posición corta (Short) — obligación de vender un instrumento, esperando que su precio baje en el futuro.

La información general sobre las operaciones comerciales está disponible en la guía de usuario del terminal de cliente..

Estas funciones pueden usarse en los Asesores Expertos y scripts. Las funciones comerciales pueden ser invocadas sólo si en las propiedades del Asesor Experto o script correspondiente está activada la opción "Permitir comerciar al Asesor Experto".

El permiso o la prohibición para tradear puede depender de muchos factores que se describen en el apartado "Permiso del trading".

Función	Acción
OrderCalcMargin	Calcula el margen requerido para el tipo de orden especificado en la moneda de depósito de la cuental
OrderCalcProfit	Calcula el beneficio basado en los parámetros pasados en la moneda de depósito de la cuenta
OrderCheck	Comprueba si la cuenta dispone de fondos suficientes para ejecutar la operación comercial requerida
OrderSend	Comprueba si hay suficientes fondos para ejecutar la operación comercial especificada.
OrderSendAsync	Envía de modo asincrónico las solicitudes comerciales sin esperar la respuesta por parte del servidor de trading
PositionsTotal	Devuelve el número de posiciones abiertas
PositionGetSymbol	Devuelve el símbolo de una posición correspondiente abierta
PositionSelect	Elige una posición abierta para el futuro trabajo con ella
PositionSelectByTicket	Selects a position to work with by the ticket number specified in it

© 2000-2025, MetaQuotes Ltd.

Función	Acción
PositionGetDouble	Devuelve la propiedad solicitada de una posición abierta (double)
PositionGetInteger	Devuelve la propiedad solicitada de una posición abierta (datetime o int)
PositionGetString	Devuelve la propiedad solicitada de una posición abierta (string)
PositionGetTicket	Returns the ticket of the position with the specified index in the list of open positions
OrdersTotal	Devuelve el número de órdenes
OrderGetTicket	Devuelve el ticket de una orden correspondiente
OrderSelect	Elige una orden para el futuro trabajo con ella
OrderGetDouble	Devuelve la propiedad solicitada de una orden (double)
OrderGetInteger	Devuelve la propiedad solicitada de una orden (datetime o int)
OrderGetString	Devuelve la propiedad solicitada de una orden (string)
HistorySelect	Solicita el historial de transacciones y órdenes del período especificado de la hora del servidor
HistoryOrderSelect	Elige en el historial una orden para el futuro trabajo con ella
HistorySelectByPosition	Solicita el historial de transacciones y órdenes con el identificador de posición especificado
HistoryOrdersTotal	Devuelve el número de órdenes en el historial
HistoryOrderGetTicket	Devuelve el ticket de una orden correspondiente en el historial
HistoryOrderGetDouble	Devuelve la propiedad solicitada de una orden en el historial (double)
HistoryOrderGetInteger	Devuelve la propiedad solicitada de una orden en el historial (datetime o int)
HistoryOrderGetString	Devuelve la propiedad solicitada de una orden en el historial (string)
HistoryDealSelect	Elige en el historial una transacción para dirigirse a ella en el futuro mediante las funciones correspondientes
HistoryDealsTotal	Devuelve el número de transacciones en el historial
HistoryDealGetTicket	Elige una transacción a procesar y devuelve el ticket de transacción en el historial
HistoryDealGetDouble	Devuelve la propiedad solicitada de una transacción en el historial (double)
HistoryDealGetInteger	Devuelve la propiedad solicitada de una transacción en el historial (datetime o int)

© 2000-2025, MetaQuotes Ltd.

Función	Acción
HistoryDealGetString	Devuelve la propiedad solicitada de una transacción en el historial (string)

© 2000-2025, MetaQuotes Ltd.

OrderCalcMargin

Calcula el margen requerido para el tipo de orden especificado , en la cuenta actual, en el entorno del mercado actual y sin tener en cuenta las órdenes pendientes actuales y posiciones abiertas. Permite evaluar el margen requerido para la operación comercial planeada. El valor se devuelve en la moneda de la cuenta.

```
bool  OrderCalcMargin(
   ENUM_ORDER_TYPE       action,           // tipo de orden
   string                symbol,           // nombre del símbolo
   double                volume,           // volumen
   double                price,            // precio de apertura
   double&               margin            // variable para la obtención del valor del
   );
```

Parámetros

action

[in] Tipo de orden, puede ser uno de los valores de la enumeración ENUM_ORDER_TYPE.

symbol

[in] Nombre del símbolo.

volume

[in] Volumen de la operación comercial a evaluar.

price

[in] Precio de apertura.

margin

[out] Variable en la que se recibirá el valor del margen requerido si la función se ejecuta con éxito. El cálculo se realiza sin tomar en consideración las órdenes pendientes ni las posiciones abiertas que pudieran haber en la cuenta actual. El valor del margen depende de muchos factores y puede ser diferente en diferentes entornos del mercado.

Valor devuelto

Devuelve true en caso de éxito. De lo contrario, la función devuelve false. Para obtener la información acerca del error, se debe usar la función GetLastError().

Ejemplo:

```
#define   VOLUME     1.0   // volumen de la orden
#define   DEVIATION  100   // distancia de colocación de la orden pendiente
#define   STOP_LIMIT 50    // distancia StopLimit de la orden

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
```

© 2000-2025, MetaQuotes Ltd.

```
   int array_types[8]={ORDER_TYPE_BUY,
                       ORDER_TYPE_SELL,
                       ORDER_TYPE_BUY_LIMIT,
                       ORDER_TYPE_SELL_LIMIT,
                       ORDER_TYPE_BUY_STOP,
                       ORDER_TYPE_SELL_STOP,
                       ORDER_TYPE_BUY_STOP_LIMIT,
                       ORDER_TYPE_SELL_STOP_LIMIT };

//--- en un ciclo por el array de los tipos de órdenes
   for(int i=0; i<(int)array_types.Size(); i++)
     {
      //--- dependiendo del tipo de la orden, obtenemos el tipo ORDER_TYPE_BUY o ORDEI
      ENUM_ORDER_TYPE type=MarketOrderByOrderType((ENUM_ORDER_TYPE)array_types[i]);

      //--- dependiendo del tipo de orden, obtenemos el precio
      double price=PriceByOrderType(_Symbol, type);

      //--- obtenemos el tamaño del margen necesario para el tipo de orden indicado en
      double margin=EMPTY_VALUE;
      ResetLastError();
      if(!OrderCalcMargin(type, _Symbol, VOLUME, price, margin))
        {
         Print("OrderCalcMargin() failed. Error ",GetLastError());
         continue;
        }
      //--- imprimimos el resultado en el registro
      PrintFormat("Margin required for %.2f %s order at price %.*f on %s symbol: %.2f
     }
   /*
   resultado:
   Margin required for 1.00 Buy order at price 1.31668 on GBPUSD symbol: 1316.68 USD
   Margin required for 1.00 Sell order at price 1.31661 on GBPUSD symbol: 1316.61 USD
   Margin required for 1.00 Buy Limit order at price 1.31568 on GBPUSD symbol: 1315.68
   Margin required for 1.00 Sell Limit order at price 1.31761 on GBPUSD symbol: 1317.6
   Margin required for 1.00 Buy Stop order at price 1.31768 on GBPUSD symbol: 1317.68
   Margin required for 1.00 Sell Stop order at price 1.31561 on GBPUSD symbol: 1315.61
   Margin required for 1.00 Buy Stop Limit order at price 1.31718 on GBPUSD symbol: 13
   Margin required for 1.00 Sell Stop Limit order at price 1.31611 on GBPUSD symbol: 1
   */
  }
//+------------------------------------------------------------------+
//| Retornamos el tipo de orden Buy o Sell según el tipo de orden    |
//+------------------------------------------------------------------+
ENUM_ORDER_TYPE MarketOrderByOrderType(ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY  : case ORDER_TYPE_BUY_LIMIT  : case ORDER_TYPE_BUY_STOP  :
```

© 2000-2025, MetaQuotes Ltd.

```
         return(ORDER_TYPE_BUY);
       case ORDER_TYPE_SELL : case ORDER_TYPE_SELL_LIMIT : case ORDER_TYPE_SELL_STOP :
         return(ORDER_TYPE_SELL);
      }
    return(WRONG_VALUE);
   }
//+------------------------------------------------------------------+
//| Retorna el tipo de precio de apertura según el tipo de orden     |
//+------------------------------------------------------------------+
double PriceByOrderType(const string symbol, const ENUM_ORDER_TYPE order_type)
   {
    int     digits=0;
    double  point=0;
    MqlTick tick={};

//--- obtenemos el valor Point del símbolo
    ResetLastError();
    if(!SymbolInfoDouble(symbol, SYMBOL_POINT, point))
      {
       Print("SymbolInfoDouble() failed. Error ", GetLastError());
       return 0;
      }

//--- obtenemos el valor Digits del símbolo
    long value=0;
    if(!SymbolInfoInteger(symbol, SYMBOL_DIGITS, value))
      {
       Print("SymbolInfoInteger() failed. Error ", GetLastError());
       return 0;
      }
    digits=(int)value;

//--- obtenemos los últimos precios del símbolo
    if(!SymbolInfoTick(symbol, tick))
      {
       Print("SymbolInfoTick() failed. Error ", GetLastError());
       return 0;
      }

//--- dependiendo del tipo de orden, retornamos el precio
    switch(order_type)
      {
       case ORDER_TYPE_BUY              :  return(tick.ask);
       case ORDER_TYPE_SELL             :  return(tick.bid);
       case ORDER_TYPE_BUY_LIMIT        :  return(NormalizeDouble(tick.ask - DEVIATION
       case ORDER_TYPE_SELL_LIMIT       :  return(NormalizeDouble(tick.bid + DEVIATION
       case ORDER_TYPE_BUY_STOP         :  return(NormalizeDouble(tick.ask + DEVIATION
       case ORDER_TYPE_SELL_STOP        :  return(NormalizeDouble(tick.bid - DEVIATION
       case ORDER_TYPE_BUY_STOP_LIMIT   :  return(NormalizeDouble(tick.ask + DEVIATION
```

© 2000-2025, MetaQuotes Ltd.

```
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return(NormalizeDouble(tick.bid - DEVIATION
      default                          :  return(0);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de orden                         |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE order_type)
  {
   switch(order_type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      case ORDER_TYPE_CLOSE_BY         :  return("Close By");
      default                          :  return("Unknown order type");
     }
  }
//+------------------------------------------------------------------+
```

Véase también

OrderSend(), Propiedades de órdenes, Tipos de operaciones comerciales

© 2000-2025, MetaQuotes Ltd.

OrderCalcProfit

Esta función calcula el beneficio para la cuenta actual en las condiciones actuales del mercado a base de los parámetros pasados. La función se utiliza para la evaluación previa de los resultados de una operación comercial. El valor se devuelve en la moneda de la cuenta.

```
bool  OrderCalcProfit(
   ENUM_ORDER_TYPE       action,           // tipo de orden (ORDER_TYPE_BUY o ORDER_TY
   string                symbol,           // nombre del símbolo
   double                volume,           // volumen
   double                price_open,       // precio de apertura
   double                price_close,      // precio de cierre
   double&               profit           // variable donde se recibe el valor del ber
   );
```

Parámetros

action

[in] Tipo de orden, puede ser uno de los valores de la enumeración ENUM_ORDER_TYPE: ORDER_TYPE_BUY o ORDER_TYPE_SELL.

symbol

[in] Nombre del símbolo.

volume

[in] Volumen de la operación comercial.

price_open

[in] Precio de apertura.

price_close

[in] Precio de cierre.

profit

[out] Variable en la que se recibirá el valor del beneficio en caso de que la función se ejecute con éxito. El valor del beneficio estimado depende de muchos factores y puede variar en diferentes entornos del mercado.

Valor devuelto

Devuelve true en caso de éxito. De lo contrario, la función devuelve false. Si se indica el tipo de orden no admisible, la función devolverá false. Para obtener la información sobre el error ocurrido, hay que llamar a la función GetLastError().

Ejemplo:

```
#define   VOLUME     1.0   // volumen de la orden
#define   DEVIATION  100   // número de puntos hasta el precio de cierre

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   string currency_profit=SymbolInfoString(_Symbol,SYMBOL_CURRENCY_PROFIT);    // divis
   double close_price=ChartPriceOnDropped(); // el precio de cierre será la coordenada

//---
   for(int i=0; i<=ORDER_TYPE_SELL; i++)
     {
      ENUM_ORDER_TYPE order_type=(ENUM_ORDER_TYPE)i;                  // tipo de orden
      double open_price=PriceOpenByOrderType(_Symbol, order_type);    // precio de aper

      //--- si el precio de apertura no ha sido obtenido, continuamos el ciclo
      if(open_price==0)
         continue;

      //--- si el precio de cierre es cero (el script no se ha iniciado arrastrándolo
      if(close_price==0)
         close_price=(order_type==ORDER_TYPE_BUY ? open_price + DEVIATION * _Point : c

      //--- calculamos la cantidad aproximada de beneficio para la cuenta actual y el
      double profit=0;
      ResetLastError();
      if(!OrderCalcProfit(order_type, _Symbol, VOLUME, open_price, close_price, profit
        {
         Print("OrderCalcProfit() failed. Error ", GetLastError());
         continue;
        }

      //--- imprimimos el valor de beneficio obtenido en el registro
      PrintFormat("Estimated profit for %.2f %s position on %s with opening price of %
                  VOLUME, OrderTypeDescription(order_type), _Symbol, _Digits, open_pri

     }
   /*
   resultado:
   Estimated profit for 1.00 Buy position on EURUSD with opening price of 1.10757 and
   Estimated profit for 1.00 Sell position on EURUSD with opening price of 1.10753 and
   */
  }
//+------------------------------------------------------------------+
//| Retorna el tipo de precio de apertura según el tipo de orden     |
//+------------------------------------------------------------------+
double PriceOpenByOrderType(const string symbol, const ENUM_ORDER_TYPE order_type)
  {
   MqlTick tick={};

//--- obtenemos los últimos precios del símbolo
   if(!SymbolInfoTick(symbol, tick))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("SymbolInfoTick() failed. Error ", GetLastError());
      return 0;
     }

//--- dependiendo del tipo de orden, retornamos el precio
   switch(order_type)
     {
      case ORDER_TYPE_BUY              :  return(tick.ask);
      case ORDER_TYPE_SELL             :  return(tick.bid);
      default                          :  return(0);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de orden                         |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE order_type)
  {
   switch(order_type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      default                          :  return("Unknown order type");
     }
  }
```

Véase también

OrderSend(), Propiedades de órdenes, Tipos de operaciones comerciales

© 2000-2025, MetaQuotes Ltd.

OrderCheck

La función OrderCheck() comprueba si la cuenta dispone de fondos suficientes para ejecutar la operación comercial requerida. Los resultados de la comprobación se colocan en los campos de la estructura MqlTradeCheckResult.

```
bool  OrderCheck(
   MqlTradeRequest&       request,      // estructura de la solicitud comercial
   MqlTradeCheckResult&   result        // estructura de la respuesta
   );
```

Parámetros

request

[in] Puntero a una estructura del tipo MqlTradeRequest que describe la acción comercial requerida.

result

[in,out] Puntero a una estructura del tipo MqlTradeCheckResult en la que se colocarán los resultados de la comprobación.

Valor devuelto

Si los fondos de la cuenta no son suficientes para la operación especificada, o los parámetros son incorrectos, la función devuelve false. Si la comprobación básica de las estructuras (comprobación de punteros) se ha realizado con éxito, la función devuelve true (**lo que no significa que la operación comercial que se solicita vaya a ser ejecutada con éxito**). Para una descripción más detallada de los resultados de la ejecución de esta función, hay que analizar los campos de la estructura *result*.

Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Ejemplo:

```
#define   DEVIATION      5              // desviación permitida del precio
#define   VOLUME         1.0            // volumen de la orden
#define   EXPERT_MAGIC   123            // MagicNumber
#define   DIRECTION      ORDER_TYPE_BUY // dirección de la orden abierta (ORDER_TYPE_BU

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos e inicializamos la estructura de la consulta, la comprobación y el re
   MqlTradeRequest      request={};
   MqlTradeCheckResult check  ={};
   MqlTradeResult       result ={};

//--- preparamos los parámetros de la consulta
   PrepareRequest(_Symbol, DIRECTION, VOLUME, request);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- preparamos los parámetros de la solicitud comercial
   ResetLastError();
   bool res=OrderCheck(request, check);
   if(!res)
     {
      PrintFormat("Trade request verification completed with error %d\nServer retcode:
      return;
     }

//--- la comprobación de la solicitud comercial se ha realizado correctamente, mostrar
   Print("Trade request verification completed successfully");
   MqlTradeCheckResultPrint(check, 14);

//--- enviamos la solicitud comercial
   if(!OrderSend(request, result))
      Print("OrderSend error ", GetLastError());    // si no hemos logrado enviar la s

//--- información sobre la operación
   PrintFormat("Trade request result: retcode=%u, deal=%I64u, order=%I64u", result.ret
   /*
   resultado con el trading automático desactivado en el terminal cliente
   Trade request verification completed with error 4752
   Server retcode: 10027, comment: AutoTrading disabled by client

   activamos el trading automático y volvemos a comprobar en el mercado cerrado:
   Experts   automated trading is enabled
   Trade request verification completed successfully
   Retcode:      0
   Balance:      10779.50 USD
   Equity:       10779.50 USD
   Profit:       0.00 USD
   Margin:       1104.79 USD
   Margin free:  9674.71 USD
   Margin level: 975.71 %
   Comment:      Done
   OrderSend error 4756
   Trade request result: retcode=10018, deal=0, order=0

   comprobamos en el mercado abierto:
   Trade request verification completed successfully
   Retcode:      0
   Balance:      10779.50 USD
   Equity:       10779.50 USD
   Profit:       0.00 USD
   Margin:       110.46 USD
   Margin free:  10669.04 USD
   Margin level: 9758.74 %
   Comment:      Done
   Trade request result: retcode=10009, deal=2777010968, order=2802818813
```

© 2000-2025, MetaQuotes Ltd.

```
   */
  }
//+------------------------------------------------------------------+
//| Comprobamos los parámetros para la solicitud comercial           |
//+------------------------------------------------------------------+
void PrepareRequest(const string symbol, const ENUM_ORDER_TYPE order_type, const doubl
  {
   ENUM_ORDER_TYPE type=(DIRECTION !=ORDER_TYPE_BUY ? ORDER_TYPE_SELL : DIRECTION);
   double price=(DIRECTION==ORDER_TYPE_BUY ? SymbolInfoDouble(Symbol(), SYMBOL_ASK) :
//--- parámetros de la solicitud
   request.action    = TRADE_ACTION_DEAL; // tipo de operación comercial
   request.symbol    = symbol;            // símbolo
   request.volume    = volume;            // volumen
   request.type      = type;              // tipo de orden
   request.price     = price;             // precio para la apertura
   request.deviation = DEVIATION;         // desviación permitida respecto al precio
   request.magic     = EXPERT_MAGIC;      // MagicNumber de la orden
  }
//+------------------------------------------------------------------+
//| Imprime en el registro los campos de la estructura               |
//| resultado de la comprobación de la solicitud comercial           |
//+------------------------------------------------------------------+
void MqlTradeCheckResultPrint(const MqlTradeCheckResult &check, const uint header_widt
  {
//--- obtenemos la divisa de la cuenta y el número de decimales para la divisa de la c
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
   int    digits  =(int)AccountInfoInteger(ACCOUNT_CURRENCY_DIGITS);

//--- definimos el texto del encabezado y la anchura del campo del encabezado
//--- si la anchura del encabezado se ha transmitido a la función como igual a cero, e
   string header="Retcode:";
   uint w=(header_width==0 ? header.Length()+1 : header_width);
//--- mostramos en el registro el código de retorno con un encabezado de la anchura es
   PrintFormat("%-*s%-u", w, header, check.retcode);

//--- mostramos en el registro el valor de balance que tendremos tras realizar la oper
   header="Balance:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.balance, currency);

//--- mostramos en el registro el valor de fondos propios que tendremos tras realizar
   header="Equity:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.equity, currency);

//--- mostramos en el registro el valor de beneficio flotante que tendremos tras reali
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.profit, currency);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos en el registro el tamaño del margen necesario para la operación comerc
   header="Margin:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.margin, currency);

//--- mostramos en el registro la cantidad de fondos propios que quedarán tras realiza
   header="Margin free:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.margin_free, currency);

//--- mostramos en el registro el nivel de margen que tendremos tras realizar  la oper
   header="Margin level:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.2f %%", w, header, check.margin_level);

//--- mostramos en el registro los comentarios al código de respuesta, la descripción
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-s", w, header, check.comment);
  }
```

Véase también

OrderSend(), Tipos de operaciones comerciales, Estructura de solicitud comercial, Estructura de comprobación de solicitud comercial, Estructura de resultado de solicitud comercial

© 2000-2025, MetaQuotes Ltd.

OrderSend

La función OrderSend() está destinada para realizar la actividad comercial dentro de los márgenes de MQL5.

```
bool  OrderSend(
   MqlTradeRequest&  request      // estructura de petición
   MqlTradeResult&   result       // estructura de respuesta
   );
```

Parámetros

request

[in] Puntero a una estructura del tipo MqlTradeRequest que describe la acción comercial del cliente.

result

[in,out] Puntero a una estructura del tipo MqlTradeResult que describe el resultado de una operación comercial en caso de llevarla a cabo con éxito (se devuelve true).

Valor devuelto

En caso de comprobar las estructuras (comprobación de punteros) con éxito, la función devuelve true, **pero eso no significa que la operación comercial que se solicita vaya a ser ejecutada con éxito**. Para una descripción más detallada de los resultados de la ejecución de esta función, hay que analizar los campos de la estructura *result*.

Nota

Las solicitudes comerciales pasan por varias fases de comprobación en el servidor comercial. En primer lugar se comprueba si todos los campos necesarios del parámetro *request* están rellenados correctamente. Si no hay errores, el servidor acepta la solicitud para su procesamiento. Si la orden se acepta con éxito, la función OrderSend() devuelve el valor true.

Se recomienda comprobar personalmente la solicitud antes de enviarla al servidor comercial. Para eso sirve la función OrderCheck() que no sólo comprueba si hay fondos suficientes en la cuenta para la ejecución de la operación comercial, sino también devuelve muchos otros parámetros útiles como resultado de comprobación de la solicitud comercial:

- código de retorno que avisa sobre un error en la solicitud que se comprueba;
- valor del balance de la cuenta que se queda tras la ejecución de la operación comercial;
- valor de fondos propios que se obtiene tras la ejecución de la operación comercial;
- valor del beneficio flotante que se obtiene tras la ejecución de la operación comercial;
- margen necesario para la operación comercial;
- fondos propios que se quedan disponibles después de realizar la operación comercial;
- nivel del margen que va a establecerse después de realizar la operación comercial;
- comentario sobre el código de respuesta, descripción del error en su caso.

Al enviar una orden de mercado (MqlTradeRequest.action=TRADE_ACTION_DEAL), si la función OrderSend()da un resultado exitoso, esto no siginifca que la orden se haya ejecutado (se han ejecutado las operaciones correspondientes): true en este caso significa solo que la orden ha sido correctamente colocada en el sistema comercial para su posterior ejecución. El servidor comercial puede rellenar los valores de los campos *deal* o *order* en la estructura del resultado retornado *result*,

© 2000-2025, MetaQuotes Ltd.

si estos datos le son conocidos en el momento de la formación de la respuesta a la llamada OrderSend(). En general, el evento o eventos de ejecución de las operaciones correspondientes a una orden pueden suceder ya después de que se envíe la respuesta a la llamada OrderSend(). Por eso, para cualquier tipo de solucitud comercial, al recibir el resultado de la ejecución de OrderSend(), es necesario en primer lugar comprobar el código de retorno del servidor comercial *retcode* y el código de respuesta del sistema comercial *retcode_external* (en caso necesario), que están disponibles en la estructura del resultado retornada *result*.

Cada orden aceptada se almacena en el servidor comercial esperando su procesamiento hasta que se de una de las condiciones adecuadas para su ejecución:

- expiración del plazo de vigencia,
- aparición de una solicitud opuesta,
- ejecución de la orden tras la recepción del precio de ejecución,
- aparición de la solicitud de cancelación de la orden.

Durante el procesamiento de la orden el servidor comercial envía un mensaje al terminal sobre el evento Trade, que se puede procesar mediante la función OnTrade().

En el servidor el resultado de ejecución de la solicitud comercial que ha sido enviada con la función OrderSend() se puede seguir a través del manejador OnTradeTransaction. Hay que tener en cuenta que durante la ejecución de una solicitud comercial el manejador OnTradeTransaction será invocado varias veces.

Por ejemplo, al enviar una orden de compra, ésta se tramita, para la cuenta se crea una orden de compra correspondiente, se realiza la ejecución de la orden, su eliminación de la lista de las abiertas, se agrega al historial de órdenes, luego la operación correspondiente se agrega al historial, y se crea una posición nueva. La función OnTradeTransaction será llamada para cada uno de estos eventos.

Ejemplo:

```
//--- valores para ORDER_MAGIC
input long order_magic=55555;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vamos a comprobar que se trata de una cuenta de demostración
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Operación del script en una cuenta real está prohibida!");
      return;
     }
//--- colocamos o eliminamos una orden
   if(GetOrdersTotalByMagic(order_magic)==0)
     {
      //--- no hay órdenes corrientes - colocamos una orden
      uint res=SendRandomPendingOrder(order_magic);
      Print("Código de devolución del servidor comercial ",res);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else // hay órdenes - eliminamos órdenes
     {
      DeleteAllOrdersByMagic(order_magic);
     }
//---
  }
//+------------------------------------------------------------------+
//|  recibir la cantidad actual de órdenes con ORDER_MAGIC especificada |
//+------------------------------------------------------------------+
int GetOrdersTotalByMagic(long const magic_number)
  {
   ulong order_ticket;
   int total=0;
//--- repasamos todas las órdenes pendientes
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
         if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
//---
   return(total);
  }
//+------------------------------------------------------------------+
//| elimina todas las órdenes pendientes con ORDER_MAGIC especifcada |
//+------------------------------------------------------------------+
void DeleteAllOrdersByMagic(long const magic_number)
  {
   ulong order_ticket;
//--- repasamos todas las órdenes pendientes
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
         //--- orden con ORDER_MAGIC apropiada
         if(magic_number==OrderGetInteger(ORDER_MAGIC))
           {
            MqlTradeResult result={};
            MqlTradeRequest request={};
            request.order=order_ticket;
            request.action=TRADE_ACTION_REMOVE;
            OrderSend(request,result);
            //--- apuntamos en el log la respuesta del servidor
            Print(__FUNCTION__,": ",result.comment," código de respuesta ",result.retc
           }
//---
  }
//+------------------------------------------------------------------+
//|  establecer una orden pendiente de una manera aleatoria          |
//+------------------------------------------------------------------+
uint SendRandomPendingOrder(long const magic_number)
  {
//--- preparamos la solicitud
   MqlTradeRequest request={};
```

© 2000-2025, MetaQuotes Ltd.

```
   request.action=TRADE_ACTION_PENDING;         // definir una orden pendiente
   request.magic=magic_number;                  // ORDER_MAGIC
   request.symbol=_Symbol;                      // instrumento
   request.volume=0.1;                          // volumen de 0.1 lote
   request.sl=0;                                // Stop Loss sin especificar
   request.tp=0;                                // Take Profit sin especificar
//--- vamos a formar el tipo de orden
   request.type=GetRandomType();                // tipo de orden
//---vamos a formar el precio para una orden pendiente
   request.price=GetRandomPrice(request.type);  // precio de apertura
//--- enviamos la orden comercial
   MqlTradeResult result={};
   OrderSend(request,result);
//--- introducimos la respuesta del servidor en el log
   Print(__FUNCTION__,":",result.comment);
   if(result.retcode==10016) Print(result.bid,result.ask,result.price);
//--- devolvemos el código de retorno del servidor comercial
   return result.retcode;
  }
//+------------------------------------------------------------------+
//| recibir el tipo de una orden pendiente de una manera aleatoria   |
//+------------------------------------------------------------------+
ENUM_ORDER_TYPE GetRandomType()
  {
   int t=MathRand()%4;
//---   0<=t<4
   switch(t)
     {
      case(0):return(ORDER_TYPE_BUY_LIMIT);
      case(1):return(ORDER_TYPE_SELL_LIMIT);
      case(2):return(ORDER_TYPE_BUY_STOP);
      case(3):return(ORDER_TYPE_SELL_STOP);
     }
//--- valor incorrecto
   return(WRONG_VALUE);
  }
//+------------------------------------------------------------------+
//|  recibir el precio de una manera aleatoria                       |
//+------------------------------------------------------------------+
double GetRandomPrice(ENUM_ORDER_TYPE type)
  {
   int t=(int)type;
//--- nivel de stops para el símbolo
   int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- recibimos los datos del último tick
   MqlTick last_tick={};
   SymbolInfoTick(_Symbol,last_tick);
//--- vamos a calcular el precio de acuerdo con el tipo
   double price;
```

© 2000-2025, MetaQuotes Ltd.

```
   if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT o ORDER_TYPE_SELL_STOP
     {
      price=last_tick.bid; // partimos del precio Bid
      price=price-(distance+(MathRand()%10)*5)*_Point;
     }
   else             // ORDER_TYPE_SELL_LIMIT o ORDER_TYPE_BUY_STOP
     {
      price=last_tick.ask; // partimos del precio Ask
      price=price+(distance+(MathRand()%10)*5)*_Point;
     }
//---
   return(price);
  }
```

Véase también

Tipos de operaciones comerciales, Estructura de solicitud comercial, Estructura de resultado de solicitud comercial

© 2000-2025, MetaQuotes Ltd.

OrderSendAsync

La función OrderSendAsync() sirve para realizar las operaciones de trading asincrónicas, sin esperar la respuesta del servidor comercial a la solicitud enviada. Esta función está diseñada para un trading de alta frecuencia, cuando según las condiciones del algoritmo de trading resulta inadmisible perder el tiempo para esperar la respuesta del servidor.

```
bool  OrderSendAsync(
   MqlTradeRequest&  request,      // estructura de la solicitud
   MqlTradeResult&   result        // estructura de la respuesta
   );
```

Parámetros

request

[in] Puntero a la estructura del tipo MqlTradeRequest que describe la acción comercial del cliente.

result

[in,out] Puntero a la estructura del tipo MqlTradeResult que describe el resultado de la operación comercial cuando la función se finaliza con éxito (si se devuelve true).

Valor devuelto

Devuelve true al enviar con éxito la solicitud comercial al servidor de trading. Si la solicitud no ha sido enviada, devuelve false. En caso de la ejecución con éxito, en la variable *result* el código de la respuesta contiene el valor TRADE_RETCODE_PLACED (código 10008) - "orden colocada". La ejecución con éxito significa sólo el hecho del envío, pero no da ninguna garantía de que la solicitud haya llegado al servidor comercial y haya sido aceptada para la tramitación. Durante el procesamiento de la solicitud recibida el servidor comercial envía al Terminal de Cliente un mensaje de respuesta referente al cambio del estado actual de la posición, órdenes y transacciones que provoca la generación del evento Trade.

En el servidor el resultado de ejecución de la solicitud comercial que ha sido enviada por la función OrderSendAsync() se puede seguir a través del manejador OnTradeTransaction. Hay que tener en cuenta que durante la ejecución de una solicitud comercial el manejador OnTradeTransaction será invocado varias veces.

Por ejemplo, al enviar una orden de compra, ésta se tramita, para la cuenta se crea una orden de compra correspondiente, se realiza la ejecución de la orden, su eliminación de la lista de las abiertas, se agrega al historial de órdenes, luego la operación correspondiente se agrega al historial, y se crea una posición nueva. La función OnTradeTransaction será llamada para cada uno de estos eventos. Para obtener la información más detallada, hay que analizar los parámetros de esta función:

- **trans** - este parámetro obtiene la estructura MqlTradeTransaction que describe la transacción comercial aplicada a la cuenta de trading;
- **request** - este parámetro obtiene la estructura MqlTradeRequest que describe la solicitud comercial que ha provocado la ejecución de la transacción comercial;
- **result** - este parámetro obtiene la estructura MqlTradeResult que describe el resultado de ejecución de la solicitud comercial.

Nota

© 2000-2025, MetaQuotes Ltd.

Esta función en términos del propósito y parámetros es igual a la OrderSend(), pero a diferencia de aquélla es una versión asincrónica. Es decir, no detiene el trabajo del programa a la espera del resultado de su ejecución. Usted puede comparar la velocidad de operaciones comerciales de estas dos funciones con la ayuda del EA del ejemplo de abajo.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#property description "EAs para el envío de solicitudes de trading "
                      " a través de la función OrderSendAsync().\r\n"
#property description "Se muestra el procesamiento de eventos de trading mediante"
                      " las funciones-manejadores OnTrade() y OnTradeTransaction()\r\n
#property description "En los parámetros del EA se puede establecer el Magic Number"
                      " (identificador único) "
#property description "y el modo de visualización de mensajes en el diario "Asesores E
//--- input parameters
input int  MagicNumber=1234567;      // Identificador del EA
input bool DescriptionModeFull=true; // Modo de visualización detallado
//--- variable para el uso en la llamada HistorySelect()
datetime history_start;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- comprobamos el permiso para el trading automático
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
     {
      Alert("El trading automático está prohibido en el terminal, el EA será eliminado
      ExpertRemove();
      return(-1);
     }
//--- no se puede tradear en la cuenta real
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("¡El EA tiene prohibido tradear en la cuenta real!");
      ExpertRemove();
      return(-2);
     }
//--- comprobar si se puede tradear en esta cuenta (por ejemplo, el trading es imposib
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
     {
      Alert("Está prohibido tradear en esta cuenta");
      ExpertRemove();
      return(-3);
     }
//--- recordamos la hora de inicio del EA para obtener el historial de trading
   history_start=TimeCurrent();
//---
   CreateBuySellButtons();
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- eliminamos los objetos gráficos
   ObjectDelete(0,"Buy");
   ObjectDelete(0,"Sell");
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- encabezamiento según el nombre de la función-manejador del evento de trading
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- obtenemos el tipo de la transacción como valor de la enumeración
   ENUM_TRADE_TRANSACTION_TYPE type=trans.type;
//--- si la transacción es el resultado del procesamiento de la solicitud
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      //---mostramos el nombre e la transacción
      Print(EnumToString(type));
      //--- luego mostramos la descripción literal de la solicitud procesada
      Print("------------RequestDescription\r\n",
            RequestDescription(request,DescriptionModeFull));
      //--- y mostramos la descripción del resultado de la solicitud
      Print("------------ ResultDescription\r\n",
            TradeResultDescription(result,DescriptionModeFull));
     }
   else // para la transacción del otro tipo mostramos la descripción completa de la t
     {
      Print("------------ TransactionDescription\r\n",
            TransactionDescription(trans,DescriptionModeFull));
     }
//---
  }
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- miembros estáticos para el almacenaje del estado de la cuenta de trading
   static int prev_positions=0,prev_orders=0,prev_deals=0,prev_history_orders=0;
//--- solicitamos el historial de trading
   bool update=HistorySelect(history_start,TimeCurrent());
   PrintFormat("HistorySelect(%s , %s) = %s",
               TimeToString(history_start),TimeToString(TimeCurrent()),(string)update);
//--- encabezamiento según el nombre de la función-manejador del evento de trading
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- mostramos el nombre del manejador y el número de órdenes para el momento de proc
   int curr_positions=PositionsTotal();
   int curr_orders=OrdersTotal();
   int curr_deals=HistoryOrdersTotal();
   int curr_history_orders=HistoryDealsTotal();
//--- mostramos el número de órdenes, posiciones, transacciones, asi como los cambios
   PrintFormat("PositionsTotal() = %d (%+d)",
               curr_positions,(curr_positions-prev_positions));
   PrintFormat("OrdersTotal() = %d (%+d)",
               curr_orders,curr_orders-prev_orders);
   PrintFormat("HistoryOrdersTotal() = %d (%+d)",
               curr_deals,curr_deals-prev_deals);
   PrintFormat("HistoryDealsTotal() = %d (%+d)",
               curr_history_orders,curr_history_orders-prev_history_orders);
//--- inserción de string break para leer el Diario con comodidad
   Print("");
//--- guardamos el estado de la cuenta
   prev_positions=curr_positions;
   prev_orders=curr_orders;
   prev_deals=curr_deals;
   prev_history_orders=curr_history_orders;
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- procesamiento del evento CHARTEVENT_CLICK ("Clic con el botón del ratón sobre el
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("=> ",__FUNCTION__,": sparam = ",sparam);
      //--- volumen mínimo para la transacción
      double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
      //--- si está pulsado el botón "Buy", compramos
      if(sparam=="Buy")
        {
         PrintFormat("Buy %s %G lot",_Symbol,volume_min);
         BuyAsync(volume_min);
         //--- despulsamos el botón pulsado
         ObjectSetInteger(0,"Buy",OBJPROP_STATE,false);
        }
      //--- si está pulsado el botón "Sell", vendemos
      if(sparam=="Sell")
        {
         PrintFormat("Sell %s %G lot",_Symbol,volume_min);
         SellAsync(volume_min);
         //--- despulsamos el botón pulsado
         ObjectSetInteger(0,"Sell",OBJPROP_STATE,false);
        }
      ChartRedraw();
     }
//---
  }
//+------------------------------------------------------------------+
//|  Devuelve la descripción literal de la transacción               |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans,
                              const bool detailed=true)
  {
//--- preparamos la cadena para el retorno desde la función
   string desc=EnumToString(trans.type)+"\r\n";
//--- en el modo detallado agregamos el máximo de información
   if(detailed)
     {
      desc+="Symbol: "+trans.symbol+"\r\n";
      desc+="Deal ticket: "+(string)trans.deal+"\r\n";
      desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
      desc+="Order ticket: "+(string)trans.order+"\r\n";
      desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
      desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
      desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
      desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
      desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
      desc+="Price trigger: "+StringFormat("%G",trans.price_trigger)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
      desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
     }
//--- devolvemos la cadena obtenida
   return desc;
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  Devuelve la descripción literal de solicitud comercial          |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request,
                          const bool detailed=true)
  {
//--- preparamos la cadena para la devolución desde la función
   string desc=EnumToString(request.action)+"\r\n";
//--- en el modo detallado agregamos el máximo de información
   if(detailed)
     {
      desc+="Symbol: "+request.symbol+"\r\n";
      desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
      desc+="Order ticket: "+(string)request.order+"\r\n";
      desc+="Order type: "+EnumToString(request.type)+"\r\n";
      desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
      desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
      desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
      desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
      desc+="Deviation points: "+StringFormat("%G",request.deviation)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
      desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
      desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
      desc+="Comment: "+request.comment+"\r\n";
     }
//--- devolvemos la cadena obtenida
   return desc;
  }
//+------------------------------------------------------------------+
//|  Devuelve la descripción literal del resultado de tramitación de la solicitud
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result,
                              const bool detailed=true)
  {
//--- preparamos la cadena para el retorno desde la función
   string desc="Retcode "+(string)result.retcode+"\r\n";
//--- en el modo detallado agregamos el máximo de información
   if(detailed)
     {
      desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
      desc+="Order ticket: "+(string)result.order+"\r\n";
      desc+="Deal ticket: "+(string)result.deal+"\r\n";
      desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
      desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
      desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
      desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
      desc+="Comment: "+result.comment+"\r\n";
     }
//--- devolvemos la cadena obtenida
   return desc;
  }
//+------------------------------------------------------------------+
//| Crea dos botones para la compra y la venta                       |
//+------------------------------------------------------------------+
void CreateBuySellButtons()
  {
//--- comprobamos la presencia del objeto con el nombre "Buy"
   if(ObjectFind(0,"Buy")>=0)
     {
      //--- si el objeto encontrado no es un botón, lo eliminamos
      if(ObjectGetInteger(0,"Buy",OBJPROP_TYPE)!=OBJ_BUTTON)
```

© 2000-2025, MetaQuotes Ltd.

```
         ObjectDelete(0,"Buy");
     }
   else
      ObjectCreate(0,"Buy",OBJ_BUTTON,0,0,0); // creamos el botón "Buy"
//--- configuramos el botón "Buy"
   ObjectSetInteger(0,"Buy",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Buy",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Buy",OBJPROP_YDISTANCE,50);
   ObjectSetInteger(0,"Buy",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Buy",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Buy",OBJPROP_TEXT,"Buy");
   ObjectSetInteger(0,"Buy",OBJPROP_COLOR,clrRed);
//--- comprobamos la presencia del objeto con el nombre "Sell"
   if(ObjectFind(0,"Sell")>=0)
     {
      //--- si el objeto encontrado no es un botón, lo eliminamos
      if(ObjectGetInteger(0,"Sell",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Sell");
     }
   else
      ObjectCreate(0,"Sell",OBJ_BUTTON,0,0,0); // creamos el botón "Sell"
//--- configuramos el botón "Sell"
   ObjectSetInteger(0,"Sell",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Sell",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Sell",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Sell",OBJPROP_TEXT,"Sell");
   ObjectSetInteger(0,"Sell",OBJPROP_COLOR,clrBlue);
//--- forzamos la actualización del gráfico para que los botones se dibujen inmediatar
   ChartRedraw();
//---
  }
//+------------------------------------------------------------------+
//| la compra mediante la función asincrónica OrderSendAsync()       |
//+------------------------------------------------------------------+
void BuyAsync(double volume)
  {
//--- preparamos la solicitud
   MqlTradeRequest req={};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_BUY;
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_ASK);
   req.deviation   =10;
   req.comment     ="Buy using OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": error ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
//| la venta mediante la función asincrónica OrderSendAsync()        |
//+------------------------------------------------------------------+
void SellAsync(double volume)
  {
//--- preparamos la solicitud
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlTradeRequest req={};
   req.action       =TRADE_ACTION_DEAL;
   req.symbol       =_Symbol;
   req.magic        =MagicNumber;
   req.volume       =0.1;
   req.type         =ORDER_TYPE_SELL;
   req.price        =SymbolInfoDouble(req.symbol,SYMBOL_BID);
   req.deviation    =10;
   req.comment      ="Sell using OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": error ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
```

Ejemplo de visualización de mensajes en el diario "Asesores Expertos":

© 2000-2025, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnChartEvent: sparam = Sell
12:52:52   ExpertAdvisor (EURUSD,H1)   Sell EURUSD 0.01 lot
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_REQUEST
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------RequestDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_ACTION_DEAL
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Magic Number: 1234567
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order filling: ORDER_FILLING_FOK
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Deviation points: 10
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Limit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Comment: Sell using OrderSendAsync()
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ ResultDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   Retcode 10009
12:52:52   ExpertAdvisor (EURUSD,H1)   Request ID: 2
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Ask: 1.29319
12:52:52   ExpertAdvisor (EURUSD,H1)   Bid: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Comment:
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+1)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_DELETE
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
```

© 2000-2025, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_HISTORY_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_FILLED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_DEAL_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
```

© 2000-2025, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
```

© 2000-2025, MetaQuotes Ltd.

PositionsTotal

Devuelve el número de posiciones abiertas.

```
int  PositionsTotal();
```

Valor devuelto

Valor del tipo int.

Nota

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos e imprimimos en el registro el número de posiciones abiertas en la cue
   int total=PositionsTotal();
   Print("Number of open positions on account: ", total);
   /*
   resultado:
   Number of open positions on account: 2
   */
  }
```

Véase también

PositionGetSymbol(), PositionSelect(), Propiedades de posiciones

© 2000-2025, MetaQuotes Ltd.

PositionGetSymbol

Devuelve el símbolo de una posición correspondiente abierta y automáticamente elige una posición para gestionarla después usando las funciones PositionGetDouble,PositionGetInteger, PositionGetString.

```
string  PositionGetSymbol(
   int  index      // número en la lista de posiciones
   );
```

Parámetros

index

[in] Número de posición en la lista de posiciones abiertas. Si la posición no ha sido encontrada, se devolverá la cadena vacía. Para obtener el código del error, hay que llamar a la función GetLastError().

Valor devuelto

Valor del tipo string. Si la posición no se ha encontrado, entonces se retornará una línea vacía. Para obtener el código de error es necesario llamar a la función GetLastError().

Nota

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos el número de posiciones abiertas en la cuenta
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el símbolo de la posición según el índice del ciclo i
      ResetLastError();
      string symbol=PositionGetSymbol(i);

      //--- si hemos obtenido el símbolo de la posición correctamente, seleccionaremos
      //--- y podemos obtener sus propiedades con la ayuda de PositionGetDouble, Posit
      if(symbol!="")
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         PrintFormat("Position symbol at index %d: %s, position type: %s", i, symbol,
        }
      else
        {
         PrintFormat("PositionGetSymbol(%d) failed. Error %d", i, GetLastError());
         continue;
        }
     }
   /*
   resultado:
   Position symbol at index 0: GBPUSD, position type: SELL
   Position symbol at index 1: EURUSD, position type: BUY
   */
  }
```

Véase también

PositionsTotal(), PositionSelect(), Propiedades de posiciones

© 2000-2025, MetaQuotes Ltd.

PositionSelect

Elige una posición abierta para trabajar posteriormente con ella. Retorna true en caso de que la función se ejecute con éxito. Retorna false en caso de que la función no se ejecute con éxito. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

```
bool  PositionSelect(
   string  symbol     // nombre del instrumento
   );
```

Parámetros

symbol

[in] Nombre del instrumento financiero.

Valor devuelto

Valor del tipo bool.

Observación

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones. En este caso, PositionSelect seleccionará la posición con el ticket menor.

La función PositionSelect() copia los datos sobre la posición en el entorno programático, y las siguientes llamadas de PositionGetDouble(), PositionGetInteger() y PositionGetString() retornarán los valores copiados anteriormente. Esto significa que la propia posición podría no existir ya (o haber modificado su volumen, dirección, etc.), pero sus datos podrían ser obtenidos todavía. Para obtener de forma garantizada datos recientes sobre una posición, se recomienda llamar a la función PositionSelect() justo antes de solicitarlos.

Ejemplo:

```
#define   SYMBOL_NAME   "EURUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- seleccionamos la posición en el símbolo indicado
   if(!PositionSelect(SYMBOL_NAME))
     {
      PrintFormat("PositionSelect(%s) failed. Error %d",SYMBOL_NAME, GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si la posición ha sido seleccionada, podemos obtener sus datos utilizando Positi
//--- obtenemos el ticket de la posición seleccionada
   ResetLastError();
   long ticket=PositionGetInteger(POSITION_TICKET);
   if(ticket==0)
     {
      PrintFormat("Failed to get %s position ticket. Error %d", SYMBOL_NAME, GetLastEr
      return;
     }

//--- si obtenemos correctamente el ticket, imprimimos el símbolo y el ticket de la po
   PrintFormat("The position that is selected on the %s symbol has ticket %I64d", SYMB
   /*
   resultado:
   The position that is selected on the EURUSD symbol has ticket 2810846623
   */
  }
```

Véase también

PositionGetSymbol(), PositionsTotal(), Propiedades de las posiciones

© 2000-2025, MetaQuotes Ltd.

PositionSelectByTicket

Elige una posición abierta para trabajar posteriormente con ella según el ticket indicado. Retorna true en caso de que la función se ejecute con éxito. Retorna false en caso de que la función no se ejecute con éxito. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

```
bool  PositionSelectByTicket(
   ulong   ticket     // ticket de la posición
   );
```

Parámetros

ticket

[in] Ticket de la posición.

Valor devuelto

Valor del tipo bool.

Observación

La función PositionSelectByTicket() copia los datos sobre la posición en el entorno programático, y las siguientes llamadas de PositionGetDouble(), PositionGetInteger() y PositionGetString() retornarán los valores copiados anteriormente. Esto significa que la propia posición podría no existir ya (o haber modificado su volumen, dirección, etc.), pero sus datos podrían ser obtenidos todavía. Para obtener de forma garantizada datos recientes sobre una posición, se recomienda llamar a la función PositionSelectByTicket() justo antes de solicitarlos.

Ejemplo:

```
#define   EXPERT_MAGIC  123456   // MagicNumber

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos e inicializamos las estructuras de solicitud y resultado
   MqlTradeRequest request={};
   MqlTradeResult  result ={};

//--- rellenamos los parámetros de la solicitud comercial para abrir una posición larg
   request.action    = TRADE_ACTION_DEAL;                       // tipo de operación co
   request.symbol    = Symbol();                                // símbolo
   request.volume    = 0.1;                                     // volumen de 0.1 lote
   request.type      = ORDER_TYPE_BUY;                          // tipo de orden
   request.price     = SymbolInfoDouble(Symbol(), SYMBOL_ASK); // precio de apertura
   request.deviation = 5;                                       // desviación permitida
   request.magic     = EXPERT_MAGIC;                            // MagicNumber de la or

//--- enviar la solicitud, si la solicitud no se ha podido enviar, mostramos el código
   if(!OrderSend(request, result))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("OrderSend error ", GetLastError());
       return;
      }

//--- mostramos la información sobre la operación
   PrintFormat("Trade request result: retcode: %u, deal: %I64u, order: %I64u", result.

//--- obtenemos el ticket de la posición a partir del resultado de la operación comerc
//--- el ticket de la posición recién abierta se corresponde con el ticket de la order
   ulong ticket=result.order;
   ResetLastError();
   if(!PositionSelectByTicket(ticket))
     {
      PrintFormat("PositionSelectByTicket(%I64u) failed. Error %d", ticket, GetLastErr
      return;
     }

//--- mostramos en el registro los datos de la posición seleccionada según el ticket
   ENUM_POSITION_TYPE type  = (ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
   long               time  = PositionGetInteger(POSITION_TIME_MSC);
   double             price = PositionGetDouble(POSITION_PRICE_OPEN);
   double             volume= PositionGetDouble(POSITION_VOLUME);
   string             symbol= PositionGetString(POSITION_SYMBOL);
   int                digits= (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   PrintFormat("Current selected position: %s %.2f %s #%I64u at %.*f, %s",
               symbol, volume, (type==POSITION_TYPE_BUY ? "Buy" : "Sell"), ticket, dig
   /*
   resultado:
   Trade request result: retcode: 10009, deal: 2778100901, order: 2803905975
   Current selected position: EURUSD 0.10 Buy #2803905975 at 1.10672, 2024.09.02 12:09
   */
  }
//+------------------------------------------------------------------+
//| Retorna la hora en milisegundos                                  |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SECO
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
  }
```

Véase también

PositionGetSymbol(), PositionsTotal(), Propiedades de las posiciones

© 2000-2025, MetaQuotes Ltd.

PositionGetDouble

La función devuelve la propiedad solicitada de una posición abierta que previamente ha sido elegida a través de la función PositionGetSymbol o PositionSelect. La propiedad de la posición tiene que ser del tipo double. Existen 2 variantes de la función.

1. Inmediatamente devuelve el valor de la propiedad.

```
double  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id      // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id,     // identificador de la propiedad
   double&                        double_var       // aquí recibimos el valor de la pr
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de la posición. Su valor puede ser uno de los valores de la enumeración ENUM_POSITION_PROPERTY_DOUBLE.

double_var

[out] Variable del tipo double que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo double.

Nota

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

Para garantizar la recepción de datos recientes sobre una posición, se recomienda llamar a la función PositionSelect() justamente antes de solicitarlos.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- en un ciclo por todas las posiciones en la cuenta
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el ticket de la siguiente posición, seleccionando una posición c
      ulong ticket=PositionGetTicket(i);
      if(ticket==0)
         continue;

      //--- obtenemos el tipo de la posición y mostramos el encabezado de la lista de
      string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" : "Sell");
      PrintFormat("Double properties of an open position %s #%I64u:", type, ticket);

      //--- imprimimos bajo el encabezado todas las propiedades reales de la posición
      PositionPropertiesDoublePrint(15);
     }
   /*
   resultado:
   Double properties of an open position Buy #2807075208:
   Volume:        1.00
   Price open:    1.10516
   StopLoss:      0.00000
   TakeProfit:    0.00000
   Price current: 1.10518
   Swap:          0.00
   Profit:        2.00 USD
   */
  }
//+------------------------------------------------------------------+
//| Muestra en el registro las propiedades reales de la posición seleccionada
//+------------------------------------------------------------------+
void PositionPropertiesDoublePrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   double value=0;

//--- obtenemos la divisa de la cuenta, el símbolo de la posición y el número de decir
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
   string symbol  =PositionGetString(POSITION_SYMBOL);
   int    digits  =(int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- definimos el texto del encabezado y la anchura del campo del encabezado
//--- si la anchura del encabezado se ha transmitido a la función como igual a cero, e
   header="Volume:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- obtenemos y mostramos en el registro el volumen de la posición con un encabezado
   if(!PositionGetDouble(POSITION_VOLUME, value))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- mostramos en el registro el valor del precio de la posición
   header="Price open:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PRICE_OPEN, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- mostramos en el regisro el valor de StopLoss
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- mostramos en el registro el valor de TakeProfit
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- mostramos en el registro el valor Price current
   header="Price current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PRICE_CURRENT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- mostramos en el registro el valor de swap acumulado
   header="Swap:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_SWAP, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- mostramos en el registro el valor de beneficio actual
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PROFIT, value))
      return;
   PrintFormat("%-*s%-.2f %s", w, header, value, currency);
  }
```

Véase también

PositionGetSymbol(), PositionSelect(), Propiedades de posiciones

© 2000-2025, MetaQuotes Ltd.

PositionGetInteger

La función devuelve la propiedad solicitada de una posición abierta que previamente ha sido elegida a través de la función PositionGetSymbol o PositionSelect. La propiedad de la posición tiene que ser del tipo datetime, int. Existen 2 variantes de la función.

1. Inmediatamente devuelve el valor de la propiedad.

```
long  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id      // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id,     // identificador de la propiedad
   long&                           long_var         // aquí recibimos el valor de la p
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de la posición. Su valor puede ser uno de los valores de la enumeración ENUM_POSITION_PROPERTY_INTEGER.

long_var

[out] Variable del tipo long que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo long. En caso de ejecución fallida devuelve 0.

Nota

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

Para garantizar la recepción de datos recientes sobre una posición, se recomienda llamar a la función PositionSelect() justamente antes de solicitarlos.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- comprobamos si la posición está presente y mostramos el tiempo de su cambio
   if(PositionSelect(_Symbol))
     {
//--- obtenemos el identificador de la posición para poder seguir trabajando con ella
      ulong position_ID=PositionGetInteger(POSITION_IDENTIFIER);
      Print(_Symbol," position #",position_ID);
//--- obtenemos el tiempo de formación de la posición en milisegundos pasados desde el
      long create_time_msc=PositionGetInteger(POSITION_TIME_MSC);
      PrintFormat("Position #%d  POSITION_TIME_MSC = %i64 milliseconds => %s",positior
                  create_time_msc,TimeToString(create_time_msc/1000));
//--- obtenemos el tiempo del último cambio de la posición en segundos desde el 01.01.
      long update_time_sec=PositionGetInteger(POSITION_TIME_UPDATE);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE = %i64 seconds => %s",
                  position_ID,update_time_sec,TimeToString(update_time_sec));
//--- obtenemos el tiempo del último cambio de la posición en milisegundos desde el 01
      long update_time_msc=PositionGetInteger(POSITION_TIME_UPDATE_MSC);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE_MSC = %i64 milliseconds => %s",
                  position_ID,update_time_msc,TimeToString(update_time_msc/1000));
     }
//---
  }
```

Véase también

PositionGetSymbol(), PositionSelect(), Propiedades de posiciones

© 2000-2025, MetaQuotes Ltd.

PositionGetString

La función devuelve la propiedad solicitada de una posición abierta que previamente ha sido elegida a través de la función PositionGetSymbol o PositionSelect. La propiedad de la posición tiene que ser del tipo string. Existen 2 variantes de la función.

1. Inmediatamente devuelve el valor de la propiedad.

```
string  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id      // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id,     // identificador de la propiedad
   string&                        string_var       // aquí recibimos el valor de la pr
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de la posición. Su valor puede ser uno de los valores de la enumeración ENUM_POSITION_PROPERTY_STRING.

string_var

[out] Variable del tipo string que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo string.

Nota

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

Para garantizar la recepción de datos recientes sobre una posición, se recomienda llamar a la función PositionSelect() justamente antes de solicitarlos.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- en un ciclo por todas las posiciones en la cuenta
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el ticket de la siguiente posición, seleccionando una posición c
      ulong ticket=PositionGetTicket(i);
      if(ticket==0)
         continue;

      //--- obtenemos el tipo de la posición y mostramos el encabezado de la lista de
      string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" : "Sell");
      PrintFormat("String properties of an open position %s #%I64u:", type, ticket);

      //--- imprimimos bajo el encabezado todas las propiedades string de la posición
      PositionPropertiesStringPrint(15);
     }
   /*
   resultado:
   String properties of an open position Buy #2810798881:
   Symbol:         EURUSD
   Comment:        Test PositionGetString
   External ID:
   */
  }
//+------------------------------------------------------------------+
//| Muestra en el registro las propiedades string de la posición seleccionada
//+------------------------------------------------------------------+
void PositionPropertiesStringPrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   string value="";

//--- definimos el texto del encabezado y la anchura del campo del encabezado
//--- si la anchura del encabezado se ha transmitido a la función como igual a cero, e
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- obtenemos y mostramos en el registro el símbolo de la posición con un encabezado
   if(!PositionGetString(POSITION_SYMBOL, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- mostramos en el registro el valor del comentario de la posición
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetString(POSITION_COMMENT, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos en el registro el valor del identificador de la posición en el sistema
   header="External ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetString(POSITION_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
```

Véase también

PositionGetSymbol(), PositionSelect(), Propiedades de posiciones

© 2000-2025, MetaQuotes Ltd.

PositionGetTicket

La función retorna el ticket de la posición según un índice de la lista de posiciones abiertas y elige de forma automática esta posición para trabajar posteriormente con ella con la ayuda de las funciones PositionGetDouble, PositionGetInteger, PositionGetString.

```
ulong  PositionGetTicket(
   int  index      // número en la lista de posiciones
   );
```

Parámetros

index

[in] Índice de la posición en la lista de posiciones abiertas, empezando por el 0.

Valor devuelto

Ticket de la posición. En caso de ejecución fallida, devuelve 0.

Observación

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

Para obtener de forma garantizada datos recientes sobre una posición, se recomienda llamar a la función PositionSelect() justo antes de solicitarlos.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- en un ciclo por todas las posiciones en la cuenta
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el ticket de la siguiente posición, seleccionando una posición c
      ulong ticket=PositionGetTicket(i);
      if(ticket==0)
         continue;

      //--- obtenemos el tipo de posición y mostramos la descripción de la posición se
      string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" : "Sell");
      PrintFormat("[%d] Selected position %s #%I64u", i, type, ticket);
```

© 2000-2025, MetaQuotes Ltd.

```
    }
  /*
 resultado:
  [0] Selected position Sell #2810802718
  [1] Selected position Buy #2810802919
  */
 }
```

Véase también

PositionGetSymbol(), PositionSelect(), Propiedades de las posiciones

© 2000-2025, MetaQuotes Ltd.

OrdersTotal

Devuelve el número de órdenes.

```
int  OrdersTotal();
```

Valor devuelto

Valor del tipo int.

Nota

No se puede confundir las órdenes pendientes con las posiciones que también se muestran en la pestaña "Operaciones" del panel "Caja de Herramientas" del terminal de cliente. Una orden es la indicación de realizar una operación comercial, mientras que una posición es el resultado de una o varias transacciones.

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos e imprimimos en el registro el número de órdenes pendientes en la cuer
   int total=OrdersTotal();
   Print("Number of active pending orders on the account: ", total);
   /*
   resultado:
   Number of active pending orders on the account: 2
   */
  }
```

Véase también

OrderSelect(), OrderGetTicket(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

OrderGetTicket

Devuelve el ticket de una orden correspondiente y automáticamente elige la orden para trabajar con ella después usando las funciones.

```
ulong  OrderGetTicket(
   int  index      // número en la lista de órdenes
   );
```

Parámetros

index

[in] Número de una orden en la lista de órdenes

Valor devuelto

Valor del tipo ulong. En caso de ejecución fallida devuelve 0.

Nota

No se puede confundir las órdenes pendientes con las posiciones que también se muestran en la pestaña "Operaciones" del panel "Caja de Herramientas" del terminal de cliente. Una orden es la indicación de realizar una operación comercial, mientras que una posición es el resultado de una o varias transacciones.

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

La función OrderGetTicket() copia los datos sobre la orden en el entorno del programa, y las posteriores llamadas OrderGetDouble(), OrderGetInteger(), OrderGetString() devuelven los datos copiados anteriormente. Eso significa que la propia orden a lo mejor ya no existe (o se ha cambiado su precio de apertura, niveles Stop Loss / Take Profit o el plazo de vencimiento), pero todavía se puede seguir recibiendo los datos sobre esta orden. Para garantizar la recepción de datos recientes sobre una orden, se recomienda llamar a la función OrderGetTicket() justamente antes de solicitarlos.

Ejemplo:

```
void OnStart()
  {
//--- variables para recibir valores desde las propiedades de la orden
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   string   symbol;
   string   type;
```

© 2000-2025, MetaQuotes Ltd.

```
   long     order_magic;
   long     positionID;
//--- número de actuales órdenes pendientes
   uint     total=OrdersTotal();
//--- repasamos todas las órdenes en el ciclo
   for(uint i=0;i<total;i++)
     {
      //--- recibimos el ticket de la orden según su posición en la lista
      if((ticket=OrderGetTicket(i))>0)
        {
         //--- recibimos las propiedades de la orden
         open_price    =OrderGetDouble(ORDER_PRICE_OPEN);
         time_setup    =(datetime)OrderGetInteger(ORDER_TIME_SETUP);
         symbol        =OrderGetString(ORDER_SYMBOL);
         order_magic   =OrderGetInteger(ORDER_MAGIC);
         positionID    =OrderGetInteger(ORDER_POSITION_ID);
         initial_volume=OrderGetDouble(ORDER_VOLUME_INITIAL);
         type          =EnumToString(ENUM_ORDER_TYPE(OrderGetInteger(ORDER_TYPE)));
         //--- preparamos y mostramos información sobre la orden
         printf("#ticket %d %s %G %s at %G was set up at %s",
                ticket,                  // ticket de la orden
                type,                    // tipo
                initial_volume,          // volumen colocado
                symbol,                  // símbolo
                open_price,              // precio de apertura especificado
                TimeToString(time_setup)// hora de colocación de la orden
                );
        }
     }
//---
  }
```

Véase también

OrdersTotal(), OrderSelect(), OrderGetInteger()

© 2000-2025, MetaQuotes Ltd.

OrderSelect

Elige una orden para el futuro trabajo con ella. Devuelve true en caso de que la ejecución de la función se finalice con éxito, de lo contrario devuelve false. Para obtener la información sobre el error hay que llamar a la función GetLastError().

```
bool  OrderSelect(
   ulong    ticket,      // ticket de la orden
   );
```

Parámetros

ticket

[in] Ticket de la orden.

Valor devuelto

Valor del tipo bool.

Nota

No se puede confundir las órdenes pendientes con las posiciones que también se muestran en la pestaña "Operaciones" del panel "Caja de Herramientas" del terminal de cliente.

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

La función OrderSelect() copia los datos sobre la orden en el entorno del programa, y las posteriores llamadas OrderGetDouble(), OrderGetInteger(), OrderGetString() devuelven los datos copiados anteriormente. Eso significa que la propia orden a lo mejor ya no existe (o se ha cambiado su precio de apertura, niveles Stop Loss / Take Profit o el plazo de vencimiento), pero todavía se puede seguir recibiendo los datos sobre esta orden. Para garantizar la recepción de datos recientes sobre una orden, se recomienda llamar a la función OrderSelect() justamente antes de solicitarlos.

Ejemplo:

```
#define   EXPERT_MAGIC  123456
#define   OFFSET        50                      // separación desde el precio actual p
#define   DIRECTION     ORDER_TYPE_BUY_LIMIT    // tipo de orden
#define   VOLUME        1.0                     // volumen
#define   DEVIATION     2                       // desviación permitida respecto al pr

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//-- declaración e inicialización de la consulta, el resultado y las variables
   MqlTradeRequest request={};
   MqlTradeResult  result ={};
   double          order_price=0;

//--- parámetros de colocación de una orden pendiente
   request.action    = TRADE_ACTION_PENDING;                                  // tipo de
   request.symbol    = _Symbol;                                               // símbolo
   request.volume    = VOLUME;                                                // volumen
   request.deviation = DEVIATION;                                             // desviaci
   request.magic     = EXPERT_MAGIC;                                          // MagicNur

//--- comprobación del tipo de operación
   switch(DIRECTION)
     {
      case ORDER_TYPE_BUY_LIMIT :
        request.type = ORDER_TYPE_BUY_LIMIT;                                  // tipo de
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_ASK)-OFFSET*_Point;// precio
        request.price= NormalizeDouble(order_price, _Digits);                 // precio n
        break;
      case ORDER_TYPE_SELL_LIMIT :
        request.type = ORDER_TYPE_SELL_LIMIT;                                 // tipo de
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_BID)+OFFSET*_Point;// precio p
        request.price= NormalizeDouble(order_price,_Digits);                  // precio n
        break;
      case ORDER_TYPE_BUY_STOP :
        request.type = ORDER_TYPE_BUY_STOP;                                   // tipo de
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_ASK)+OFFSET*_Point;// precio p
        request.price= NormalizeDouble(order_price,_Digits);                  // precio n
        break;
      case ORDER_TYPE_SELL_STOP :
        request.type = ORDER_TYPE_SELL_STOP;                                  // tipo de
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_BID)-OFFSET*_Point;// precio p
        request.price= NormalizeDouble(order_price,_Digits);                  // precio n
        break;
      default: // si no se ha elegido una orden pendiente o StopLimit
        Alert("This example is only for placing pending orders BuyLimit, SellLimit, Bu
        break;
     }

//--- enviar la solicitud, si la solicitud no se ha podido enviar, mostramos el código
   if(!OrderSend(request, result))
     {
      Print("OrderSend error ", GetLastError());
      return;
     }

//--- mostramos la información sobre la operación
   PrintFormat("Trade request result: retcode=%u, order=%I64u", result.retcode, result
```

© 2000-2025, MetaQuotes Ltd.

```
//--- obtenemos el ticket de la orden a partir del resultado de la operación comercial
   ulong ticket=result.order;
   ResetLastError();
   if(!OrderSelect(ticket))
     {
      PrintFormat("OrderSelect(%I64u) failed. Error %d", ticket, GetLastError());
      return;
     }

//--- mostramos en el registro los datos de la orden seleccionada según el ticket
   ENUM_ORDER_TYPE   type  = (ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE);
   long              time  = OrderGetInteger(ORDER_TIME_SETUP_MSC);
   double            price = OrderGetDouble(ORDER_PRICE_OPEN);
   double            volume= OrderGetDouble(ORDER_VOLUME_CURRENT);
   string            symbol= OrderGetString(ORDER_SYMBOL);
   int               digits= (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   PrintFormat("Current selected order: %s %.2f %s #%I64u at %.*f, %s",
               symbol, volume, OrderTypeDescription(type), ticket, digits, price, Time
   /*
   resultado:
   Trade request result: retcode=10009, order=2811006719
   Current selected order: EURUSD 1.00 Buy Limit #2811006719 at 1.10550, 2024.09.04 10
   */
  }
//+------------------------------------------------------------------+
//| Retorna la hora en milisegundos                                  |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de orden                         |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type");
     }
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

Véase también

OrderGetInteger(), OrderGetDouble(), OrderGetString(), OrderCalcProfit(), OrderGetTicket(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

OrderGetDouble

La función devuelve la propiedad solicitada de una orden que previamente ha sido elegida a través de la función OrderGetTicket o OrderSelect. La propiedad de la orden tiene que ser del tipo double. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
double  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id        // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,      // identificador de la propiedad
   double&                     double_var        // aquí recibimos el valor de la prop
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de la orden. Su valor puede ser uno de los valores de la enumeración ENUM_ORDER_PROPERTY_DOUBLE.

double_var

[out] Variable del tipo double que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo double.

Nota

No se puede confundir las órdenes pendientes con las posiciones que también se muestran en la pestaña "Operaciones" del panel "Caja de Herramientas" del terminal de cliente.

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

Para garantizar la recepción de datos recientes sobre una orden, se recomienda llamar a la función OrderSelect() justamente antes de solicitarlos.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- en un ciclo por la lista de todas las órdenes en la cuenta
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el ticket de la orden en la lista según el índice del ciclo
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- obtenemos el tipo de la orden y mostramos el encabezado de la lista de pro
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE));
      PrintFormat("Double properties of an active pending order %s #%I64u:", type, tic

      //--- imprimimos bajo el encabezado todas las propiedades reales de la orden sel
      OrderPropertiesDoublePrint(16);
     }
   /*
   resultado:
   Double properties of an active pending order Sell Limit #2812000714:
   Volume initial: 1.00
   Volume current: 1.00
   Price open:     145.282
   StopLoss:       0.000
   TakeProfit:     0.000
   Price current:  145.044
   StopLimit:      0.000
   Double properties of an active pending order Buy Limit #2812001112:
   Volume initial: 1.00
   Volume current: 1.00
   Price open:     144.836
   StopLoss:       0.000
   TakeProfit:     0.000
   Price current:  145.051
   StopLimit:      0.000
   Double properties of an active pending order Buy Stop #2812001488:
   Volume initial: 0.50
   Volume current: 0.50
   Price open:     1.10642
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10530
   StopLimit:      0.00000
   Double properties of an active pending order Sell Stop #2812001712:
   Volume initial: 0.50
   Volume current: 0.50
   Price open:     1.10374
```

© 2000-2025, MetaQuotes Ltd.

```
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10525
   StopLimit:      0.00000
   */
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de orden                         |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type");
     }
  }
//+------------------------------------------------------------------+
//| Muestra en el registro las propiedades reales de la orden elegida|
//+------------------------------------------------------------------+
void OrderPropertiesDoublePrint(const uint header_width=0)
  {
//--- obtenemos el símbolo de la orden y el número de decimales del símbolo
   string symbol = OrderGetString(ORDER_SYMBOL);
   int    digits = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- mostramos en el registro el volumen inicial al colocar la orden con un encabezac
   OrderPropertyPrint("Volume initial:",header_width,2,ORDER_VOLUME_INITIAL);

//--- mostramos en el registro el valor del volumen no ejecutado de la orden
   OrderPropertyPrint("Volume current:",header_width,2,ORDER_VOLUME_CURRENT);

//--- mostramos en el registro el valor del precio indicado en la orden
   OrderPropertyPrint("Price open:",header_width,digits,ORDER_PRICE_OPEN);

//--- mostramos en el registro el valor del nivel de StopLoss
   OrderPropertyPrint("StopLoss:",header_width,digits,ORDER_SL);

//--- mostramos en el registro el valor del nivel de TakeProfit
   OrderPropertyPrint("TakeProfit:",header_width,digits,ORDER_TP);

//--- mostramos en el registro el valor del precio actual según el símbolo de la order
```

© 2000-2025, MetaQuotes Ltd.

```
   OrderPropertyPrint("Price current:",header_width,digits,ORDER_PRICE_CURRENT);

//--- mostramos en el registro el valor de precio de la colocación de la orden Limit a
   OrderPropertyPrint("StopLimit:",header_width,digits,ORDER_PRICE_STOPLIMIT);
  }
//+------------------------------------------------------------------+
//| Muestra en el registro una propiedad real de la orden            |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width, int digits, ENUM_ORDER
  {
   double value=0;
   if(!OrderGetDouble(property, value))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property), GetLastE
   else
     {
      //--- si la anchura del encabezado se ha transmitido a la función como igual a 0
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-.*f", w, header, digits, value);
     }
  }
```

Véase también

OrdersTotal(), OrderGetTicket(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

OrderGetInteger

La función devuelve la propiedad solicitada de una orden que previamente ha sido elegida a través de la función OrderGetTicket o OrderSelect. La propiedad de la orden tiene que ser del tipo datetime, int. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
long  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id,      // identificador de la propiedad
   long&                        long_var          // aquí recibimos el valor de la pro
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de la orden. Su valor puede ser uno de los valores de la enumeración ENUM_ORDER_PROPERTY_INTEGER.

long_var

[out] Variable del tipo long que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo long.

Nota

No se puede confundir las órdenes pendientes con las posiciones que también se muestran en la pestaña "Operaciones" del panel "Caja de Herramientas" del terminal de cliente.

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

Para garantizar la recepción de datos recientes sobre una orden, se recomienda llamar a la función OrderSelect() justamente antes de solicitarlos.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- en un ciclo por la lista de todas las órdenes en la cuenta
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el ticket de la orden en la lista según el índice del ciclo
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- obtenemos el tipo de la orden y mostramos el encabezado de la lista de pro
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE));
      PrintFormat("Integer properties of an active pending order %s #%I64u:", type, ti

      //--- imprimimos bajo el encabezado todas las propiedades enteras de la orden se
      OrderPropertiesIntegerPrint(17);
     }
   /*
  resultado:
   Integer properties of an active pending order Buy Limit #2812945317:
   Ticket:           2812945317
   Time setup:       2024.09.04 19:17:16
   Type:             Buy Limit
   State:            Placed
   Time expiration:  0
   Time done:        0
   Time setup msc:   2024.09.04 19:17:16.686
   Time done msc:    0
   Type filling:     Return
   Type time:        Time GTC
   Magic:            0
   Reason:           Client
   Position ID:      0
   Position By ID:   0
   */
  }
//+------------------------------------------------------------------+
//| Muestra en el registro las propiedades reales de la orden elegida
//+------------------------------------------------------------------+
void OrderPropertiesIntegerPrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   long   value=0;

//--- mostramos en el registro el ticket de la orden
   OrderPropertyPrint("Ticket:", header_width, ORDER_TICKET);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos en el registro el valor de hora de ejecución de la orden
   OrderPropertyPrint("Time setup:", header_width, ORDER_TIME_SETUP);

//--- mostramos en el registro el tipo de la orden
   OrderPropertyPrint("Type:", header_width, ORDER_TYPE);

//--- mostramos en el registro el estado de la orden
   OrderPropertyPrint("State:", header_width, ORDER_STATE);

//--- mostramos en el registro la hora de expiración de la orden
   OrderPropertyPrint("Time expiration:", header_width, ORDER_TIME_EXPIRATION);

//--- mostramos en el registro la hora de ejecución o retirada de la orden
   OrderPropertyPrint("Time done:", header_width, ORDER_TIME_DONE);

//--- mostramos en el registro la hora de colocación de la orden para su ejecución en
   OrderPropertyPrint("Time setup msc:", header_width, ORDER_TIME_SETUP_MSC);

//--- mostramos en el registro la hora de ejecución o retirada de la orden en miliseng
   OrderPropertyPrint("Time done msc:", header_width, ORDER_TIME_DONE_MSC);

//--- mostramos en el registro el tipo de ejecución según el resto
   OrderPropertyPrint("Type filling:", header_width, ORDER_TYPE_FILLING);

//--- mostramos en el registro el tiempo de vida útil de la orden
   OrderPropertyPrint("Type time:", header_width, ORDER_TYPE_TIME);

//--- mostramos en el registro el identificador del experto que ha colocado la orden
   OrderPropertyPrint("Magic:", header_width, ORDER_MAGIC);

//--- mostramos en el registro el motivo o fuente de la colocación de la orden
   OrderPropertyPrint("Reason:", header_width, ORDER_REASON);

//--- mostramos en el registro el identificador de la posición colocado en la orden al
   OrderPropertyPrint("Position ID:", header_width, ORDER_POSITION_ID);

//--- mostramos en el registro el identificador de la posición opuesta para las órdene
   OrderPropertyPrint("Position By ID:", header_width, ORDER_POSITION_BY_ID);
  }
//+------------------------------------------------------------------+
//| Muestra en el registro el valor de propiedad entera de la orden  |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width, ENUM_ORDER_PROPERTY_IN
  {
   string svalue="";
   long   lvalue=0;
   if(!OrderGetInteger(property, lvalue))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property), GetLastE
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      switch(property)
        {
         case ORDER_TICKET          :
         case ORDER_MAGIC           :
         case ORDER_POSITION_ID     :
         case ORDER_POSITION_BY_ID  :
           svalue=(string)lvalue;
           break;

         case ORDER_TIME_SETUP      :
         case ORDER_TIME_EXPIRATION :
         case ORDER_TIME_DONE       :
           svalue=(lvalue!=0 ? TimeToString((datetime)lvalue, TIME_DATE|TIME_MINUTES|T
           break;

         case ORDER_TIME_SETUP_MSC  :
         case ORDER_TIME_DONE_MSC   :
           svalue=(lvalue!=0 ? TimeMscToString(lvalue) : "0");
           break;

         case ORDER_TYPE            :
           svalue=OrderTypeDescription((ENUM_ORDER_TYPE)lvalue);
           break;
         case ORDER_STATE           :
           svalue=OrderStateDescription((ENUM_ORDER_STATE)lvalue);
           break;
         case ORDER_TYPE_FILLING    :
           svalue=OrderTypeFillingDescription((ENUM_ORDER_TYPE_FILLING)lvalue);
           break;
         case ORDER_TYPE_TIME       :
           svalue=OrderTypeTimeDescription((ENUM_ORDER_TYPE_TIME)lvalue);
           break;
         case ORDER_REASON          :
           svalue=OrderReasonDescription((ENUM_ORDER_REASON)lvalue);
           break;

         default                    :
           svalue="Unknown property";
           break;
        }

      //--- si la anchura del encabezado se ha transmitido a la función como igual a c
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-s", w, header, svalue);
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Retorna la descripción del tipo de orden                         |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del estado de la orden                    |
//+------------------------------------------------------------------+
string OrderStateDescription(ENUM_ORDER_STATE state)
  {
   switch(state)
     {
      case ORDER_STATE_STARTED         :  return("Started");
      case ORDER_STATE_PLACED          :  return("Placed");
      case ORDER_STATE_CANCELED        :  return("Canceled");
      case ORDER_STATE_PARTIAL         :  return("Partial");
      case ORDER_STATE_FILLED          :  return("Filled");
      case ORDER_STATE_REJECTED        :  return("Rejected");
      case ORDER_STATE_EXPIRED         :  return("Expired");
      case ORDER_STATE_REQUEST_ADD     :  return("Request Add");
      case ORDER_STATE_REQUEST_MODIFY  :  return("Request Modify");
      case ORDER_STATE_REQUEST_CANCEL  :  return("Request Cancel");
      default                          :  return("Unknown state: "+(string)state);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción de la política de ejecución del volumen de la orden
//+------------------------------------------------------------------+
string OrderTypeFillingDescription(const ENUM_ORDER_TYPE_FILLING type)
  {
   switch(type)
     {
      case ORDER_FILLING_FOK     :  return("Fill or Kill");
      case ORDER_FILLING_IOC     :  return("Immediate or Cancel");
      case ORDER_FILLING_BOC     :  return("Book or Cancel");
      case ORDER_FILLING_RETURN  :  return("Return");
      default                    :  return("Unknown type filling: "+(string)type);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
//+------------------------------------------------------------------+
//| Retorna la descripción de la línea de acción de la orden         |
//+------------------------------------------------------------------+
string OrderTypeTimeDescription(const ENUM_ORDER_TYPE_TIME type)
   {
    switch(type)
      {
       case ORDER_TIME_GTC           :   return("Time GTC");
       case ORDER_TIME_DAY           :   return("Time Day");
       case ORDER_TIME_SPECIFIED     :   return("Time Specified");
       case ORDER_TIME_SPECIFIED_DAY :   return("Time Specified Day");
       default                       :  return("Unknown type time: "+(string)type);
      }
   }
//+------------------------------------------------------------------+
//| Retorna la descripción del motivo de la colocación de la orden   |
//+------------------------------------------------------------------+
string OrderReasonDescription(const ENUM_ORDER_REASON reason)
   {
    switch(reason)
      {
       case ORDER_REASON_CLIENT    :  return("Client");
       case ORDER_REASON_MOBILE    :  return("Mobile");
       case ORDER_REASON_WEB       :  return("Web");
       case ORDER_REASON_EXPERT    :  return("Expert");
       case ORDER_REASON_SL        :  return("Stop Loss");
       case ORDER_REASON_TP        :  return("Take Profit");
       case ORDER_REASON_SO        :  return("Stop Out");
       default                     :  return("Unknown reason: "+(string)reason);
      }
   }
//+------------------------------------------------------------------+
//| Retorna la hora en milisegundos                                  |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
   {
    return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
   }
```

Véase también

OrdersTotal(), OrderGetTicket(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

OrderGetString

La función devuelve la propiedad solicitada de una orden que previamente ha sido elegida a través de la función OrderGetTicket o OrderSelect. La propiedad de la orden tiene que ser del tipo string. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
string  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id        // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id,      // identificador de la propiedad
   string&                     string_var        // aquí recibimos el valor de la prop
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de la orden. Su valor puede ser uno de los valores de la enumeración ENUM_ORDER_PROPERTY_STRING.

string_var

[out] Variable del tipo strng que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo string.

Nota

No se puede confundir las órdenes pendientes con las posiciones que también se muestran en la pestaña "Operaciones" del panel "Caja de Herramientas" del terminal de cliente.

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones.

Para garantizar la recepción de datos recientes sobre una orden, se recomienda llamar a la función OrderSelect() justamente antes de solicitarlos.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- en un ciclo por la lista de todas las órdenes en la cuenta
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el ticket de la orden en la lista según el índice del ciclo
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- obtenemos el tipo de la orden y mostramos el encabezado de la lista de pro
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE));
      PrintFormat("String properties of an active pending order %s #%I64u:", type, tic

      //--- imprimimos bajo el encabezado todas las propiedades string de la orden sel
      OrderPropertiesStringPrint(13);
     }
   /*
   resultado:
   String properties of an active pending order Sell Limit #2813781342:
   Comment:     Test OrderGetString
   Symbol:      EURUSD
   External ID:
   */
  }
//+------------------------------------------------------------------+
//| Muestra en el registro las propiedades string de la orden elegida|
//+------------------------------------------------------------------+
void OrderPropertiesStringPrint(const uint header_width=0)
  {
//--- mostramos en el registro los comentarios
   OrderPropertyPrint("Comment:", header_width, ORDER_COMMENT);

//--- mostramos en el registro el símbolo del que se ha colocado la orden
   OrderPropertyPrint("Symbol:", header_width, ORDER_SYMBOL);

//--- mostramos en el registro el identificador de la orden en el sistema comercial ex
   OrderPropertyPrint("External ID:", header_width, ORDER_EXTERNAL_ID);
  }
//+------------------------------------------------------------------+
//| Muestra en el registro el valor de la propiedad string de la orden              |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width, ENUM_ORDER_PROPERTY_ST
  {
   string value="";
   if(!OrderGetString(property, value))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property), GetLastE
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- si la anchura del encabezado se ha transmitido a la función como igual a 0
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-s", w, header, value);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de orden                         |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
```

Véase también

OrdersTotal(), OrderGetTicket(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

HistorySelect

Solicita el historial de transacciones y órdenes del período especificado de la hora del servidor.

```
bool  HistorySelect(
   datetime  from_date,      // desde la fecha
   datetime  to_date         // hasta la fecha
   );
```

Parámetros

from_date

[in] Fecha inicial de solicitud.

to_date

[in] Fecha final de solicitud.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

La función HistorySelect() crea en el programa mql5 la lista de órdenes y la de transacciones para recurrir posteriormente a estos elementos a través de las funciones correspondientes. El tamaño de la lista de transacciones se puede saber usando la función HistoryDealsTotal(), el tamaño de la lista de órdenes en el historial se puede averiguar con HistoryOrdersTotal(). La selección de los elementos de la lista de órdenes es mejor realizar usando la función HistoryOrderGetTicket(), para los elementos de la lista de transacciones mejor conviene la función HistoryDealGetTicket().

Después de usar la función HistoryOrderSelect() la lista de órdenes en el historial, que están disponibles para el programa mql5, se pone a cero y se llena de nuevo con la orden encontrada, si la búsqueda de orden por el ticket ha sido completada con éxito. Lo mismo se refiere a la lista de transacciones disponibles en el programa mql5, es decir, se pone a cero con la función HistoryDealSelect() y vuelve a llenarse en caso de recibir con éxito una transacción por el número del ticket.

Ejemplo:

```
void OnStart()
  {
   color BuyColor =clrBlue;
   color SellColor=clrRed;
//--- request trade history
   HistorySelect(0,TimeCurrent());
//--- create objects
   string   name;
   uint     total=HistoryDealsTotal();
   ulong    ticket=0;
   double   price;
   double   profit;
   datetime time;
   string   symbol;
```

© 2000-2025, MetaQuotes Ltd.

```
   long     type;
   long     entry;
//--- for all deals
   for(uint i=0;i<total;i++)
     {
      //--- try to get deals ticket
      if((ticket=HistoryDealGetTicket(i))>0)
        {
         //--- get deals properties
         price =HistoryDealGetDouble(ticket,DEAL_PRICE);
         time  =(datetime)HistoryDealGetInteger(ticket,DEAL_TIME);
         symbol=HistoryDealGetString(ticket,DEAL_SYMBOL);
         type  =HistoryDealGetInteger(ticket,DEAL_TYPE);
         entry =HistoryDealGetInteger(ticket,DEAL_ENTRY);
         profit=HistoryDealGetDouble(ticket,DEAL_PROFIT);
         //--- only for current symbol
         if(price && time && symbol==Symbol())
           {
            //--- create price object
            name="TradeHistory_Deal_"+string(ticket);
            if(entry) ObjectCreate(0,name,OBJ_ARROW_RIGHT_PRICE,0,time,price,0,0);
            else      ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,time,price,0,0);
            //--- set object properties
            ObjectSetInteger(0,name,OBJPROP_SELECTABLE,0);
            ObjectSetInteger(0,name,OBJPROP_BACK,0);
            ObjectSetInteger(0,name,OBJPROP_COLOR,type?BuyColor:SellColor);
            if(profit!=0) ObjectSetString(0,name,OBJPROP_TEXT,"Profit: "+string(profit
           }
        }
     }
//--- apply on chart
   ChartRedraw();
  }
```

Véase también

HistoryOrderSelect(), HistoryDealSelect()

© 2000-2025, MetaQuotes Ltd.

HistorySelectByPosition

Busca el historial de transacciones y órdenes que tienen el identificador de posición especificado.

```
bool  HistorySelectByPosition(
   ulong   position_id     // identificador de posición - POSITION_IDENTIFIER
   );
```

Parámetros

position_id

[in] Identificador de posición que se asigna a cada orden ejecutada y en cada transacción.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

No se debe confundir las órdenes del historial de trading con las órdenes pendientes actuales que se muestran en la pestaña "Trading" del panel "Caja de Herramientas" del terminal de cliente. La lista de las órdenes que han sido canceladas o las que han llevado a la ejecución de una operación comercial se puede ver en la pestaña "Historial" del panel "Caja de Herramientas" del terminal de cliente.

La función HistorySelectByPosition() crea la lista de órdenes y la lista de transacciones con el identificador de posición especificado para recurrir posteriormente a estos elementos a través de las funciones correspondientes. El tamaño de la lista de transacciones se puede saber usando la función HistoryDealsTotal(), el tamaño de la lista de órdenes en el historial se puede obtener con HistoryOrdersTotal(). La selección de los elementos de la lista de órdenes es mejor realizar usando la función HistoryOrderGetTicket(), para los elementos de la lista de transacciones mejor conviene la función HistoryDealGetTicket().

Después de usar la función HistoryOrderSelect() la lista de órdenes en el historial, disponibles para el programa mql5, se pone a cero y se llena de nuevo con la orden encontrada, si la búsqueda de orden por el ticket ha sido completada con éxito. Lo mismo se refiere a la lista de transacciones disponibles en el programa mql5, es decir, se pone a cero con la función HistoryDealSelect() y vuelve a llenarse en caso de recibir con éxito una transacción por el número del ticket.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   long  pos_id_array[];         // array para guardar el identificador de las posici

//--- solicitamos la historia completa
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- recopilamos en un array todos los ID de posición que surgen solo de órdenes pend
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
     {
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket, ORDER_TYPE)
      long pos_id=HistoryOrderGetInteger(ticket, ORDER_POSITION_ID);
      if(type<=ORDER_TYPE_SELL || pos_id==0)
         continue;

      int size=ArraySize(pos_id_array);
      if(ArrayResize(pos_id_array, size+1)==size+1)
         pos_id_array[size]=pos_id;
     }

//--- según la lista de identificadores de posiciones en el array
   total=ArraySize(pos_id_array);
   for(int i=0; i<total; i++)
     {
      //--- imprimimos en el registro el encabezado y la lista de órdenes y operacione
      long position_id=pos_id_array[i];
      Print("List of orders and deals for position with ID: ", position_id);
      HistorySelectByPositionProcess(position_id);
     }
   /*
   resultado:
   List of orders and deals for position with ID: 1819629924
     [0] Order Sell Limit #1819629924
     [1] Order Buy #1819633194
     [0] Entry In Deal Sell #1794972472
     [1] Entry Out Deal Buy #1794975589
   List of orders and deals for position with ID: 1841753970
     [0] Order Sell Stop #1841753970
     [1] Order Buy #1842322160
     [0] Entry In Deal Sell #1817242142
     [1] Entry Out Deal Buy #1817765341
   */
  }
//+------------------------------------------------------------------+
//| Elige la historia de órdenes y operaciones según el ID de la posición e
//| imprime en el registro la lista de órdenes y operaciones de la posición
//+------------------------------------------------------------------+
bool HistorySelectByPositionProcess(const long position_id)
  {
//--- solicitamos la historia de operaciones y órdenes con el ID de posición especific
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!HistorySelectByPosition(position_id))
     {
      PrintFormat("HistorySelectByPosition(%I64d) failed. Error %d", position_id, GetI
      return(false);
     }

//--- imprimimos en el registro la lista de órdenes de la posición
   int orders_total=HistoryOrdersTotal();
   for(int i=0; i<orders_total; i++)
     {
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;
      ENUM_ORDER_TYPE order_type=(ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket, ORDEI
      PrintFormat("  [%d] Order %s #%I64u", i, OrderTypeDescription(order_type), ticke
     }

//--- imprimimos en el registro la lista de operaciones de la posición
   int deals_total =HistoryDealsTotal();
   for(int i=0; i<deals_total; i++)
     {
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;
      ENUM_DEAL_ENTRY deal_entry=(ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket, DEAL_E
      ENUM_DEAL_TYPE  deal_type= (ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket, DEAL_TY
      if(deal_type!=DEAL_TYPE_BUY && deal_type!=DEAL_TYPE_SELL)
         continue;
      PrintFormat("  [%d] Entry %s Deal %s #%I64u", i, DealEntryDescription(deal_entry
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de orden                         |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)type);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de operación de la posición      |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      //--- retornamos la descripción solo de los tipos de operaciones Buy y Sell,
      //--- ya que todos los demás tipos no pertenecen a la posición
      case DEAL_TYPE_BUY    :  return("Buy");
      case DEAL_TYPE_SELL   :  return("Sell");
      default               :  return("Unknown deal type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del método de cambio de posición          |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("InOut");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
```

See also

HistorySelect(), HistoryOrderGetTicket(), Order Properties

© 2000-2025, MetaQuotes Ltd.

HistoryOrderSelect

Elige en el historial una orden para recurrir posteriormente a ésta a través de las funciones correspondientes. Devuelve true en caso de que la ejecución de la función se finalice con éxito, de lo contrario devuelve false. Para obtener la información sobre el error hay que llamar a la función GetLastError().

```
bool  HistoryOrderSelect(
   ulong  ticket,      // ticket de orden
   );
```

Parámetros

ticket

[in] Ticket de la orden.

Valor devuelto

Devuelve true en caso del éxito, de lo contrario devuelve false.

Nota

No se debe confundir las órdenes del historial de trading con las órdenes pendientes actuales que se muestran en la pestaña "Trading" del panel "Caja de Herramientas" del terminal de cliente. La lista de las órdenes que han sido canceladas o las que han llevado a la ejecución de una operación comercial se puede ver en la pestaña "Historial" del panel "Caja de Herramientas" del terminal de cliente.

La función HistoryOrderSelect() limpia en un programa mql5 la lista de órdenes del historial, las que están disponible, y copia en esta lista sólo una orden, si la ejecución de la función HistoryOrderSelect() se ha completado con éxito. Si hace falta repasar todas las transacciones seleccionadas por la función HistorySelect() es mejor utilizar la función HistoryOrderGetTicket().

Ejemplo:

```
#define   TICKET    1819621374   // ticket de cualquier orden conocida, por ejemplo, c
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- seleccionamos una orden histórica según el ticket especificado en TICKET
   if(!HistoryOrderSelect(TICKET))
     {
      PrintFormat("HistoryOrderSelect(%I64u) failed. Error %d", TICKET, GetLastError()
      return;
     }

//--- si la orden se ha seleccionado con éxito, obtenemos sus datos y mostramos la des
   ENUM_ORDER_TYPE   order_type  = (ENUM_ORDER_TYPE)HistoryOrderGetInteger(TICKET, ORI
   ENUM_ORDER_STATE  order_state = (ENUM_ORDER_STATE)HistoryOrderGetInteger(TICKET, OR
   ENUM_ORDER_REASON order_reason= (ENUM_ORDER_REASON)HistoryOrderGetInteger(TICKET, O
   long              order_time  = HistoryOrderGetInteger(TICKET, ORDER_TIME_SETUP_MSC
```

© 2000-2025, MetaQuotes Ltd.

```
   string            order_symbol=HistoryOrderGetString(TICKET, ORDER_SYMBOL);
   double            order_vol_init=HistoryOrderGetDouble(TICKET, ORDER_VOLUME_INITIAL
   double            order_vol_curr=HistoryOrderGetDouble(TICKET, ORDER_VOLUME_CURRENT
   PrintFormat("%s Order %.2f/%s %s #%I64u %s by %s at %s",
               order_symbol, order_vol_init, (order_vol_curr>0 ? DoubleToString(order_
               OrderTypeDescription(order_type), TICKET, OrderStateDescription(order_s
               OrderReasonDescription(order_reason), TimeMscToString(order_time));
   /*
   resultado para los diferentes tickets indicados:
   EURUSD Order 0.50/0.50 Buy Limit #2812894647 Canceled by Client at 2024.09.04 19:02
   EURUSD Order 0.10/0 Sell #1753011743 Filled by Take Profit at 2023.06.12 17:04:20.3
   GBPUSD Order 0.10/0 Buy #1819621374 Filled by Client at 2023.07.24 06:16:25.746
   */
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de orden                         |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del estado de la orden                    |
//+------------------------------------------------------------------+
string OrderStateDescription(ENUM_ORDER_STATE state)
  {
   switch(state)
     {
      case ORDER_STATE_STARTED         :  return("Started");
      case ORDER_STATE_PLACED          :  return("Placed");
      case ORDER_STATE_CANCELED        :  return("Canceled");
      case ORDER_STATE_PARTIAL         :  return("Partial");
      case ORDER_STATE_FILLED          :  return("Filled");
      case ORDER_STATE_REJECTED        :  return("Rejected");
      case ORDER_STATE_EXPIRED         :  return("Expired");
      case ORDER_STATE_REQUEST_ADD     :  return("Request Add");
      case ORDER_STATE_REQUEST_MODIFY  :  return("Request Modify");
      case ORDER_STATE_REQUEST_CANCEL  :  return("Request Cancel");
```

© 2000-2025, MetaQuotes Ltd.

```
      default                          :  return("Unknown state: "+(string)state);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del motivo de la colocación de la orden   |
//+------------------------------------------------------------------+
string OrderReasonDescription(const ENUM_ORDER_REASON reason)
  {
   switch(reason)
     {
      case ORDER_REASON_CLIENT    :  return("Client");
      case ORDER_REASON_MOBILE    :  return("Mobile");
      case ORDER_REASON_WEB       :  return("Web");
      case ORDER_REASON_EXPERT    :  return("Expert");
      case ORDER_REASON_SL        :  return("Stop Loss");
      case ORDER_REASON_TP        :  return("Take Profit");
      case ORDER_REASON_SO        :  return("Stop Out");
      default                     :  return("Unknown reason: "+(string)reason);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la hora en milisegundos                                  |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
  }
```

Véase también

HistorySelect(), HistoryOrderGetTicket(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

HistoryOrdersTotal

Devuelve el número de órdenes en el historial. Antes de llamar a la función HistoryOrdersTotal(), hay que recibir el historial de transacciones y órdenes usando la función HistorySelect() o HistorySelectByPosition().

```
int  HistoryOrdersTotal();
```

Valor devuelto

Valor del tipo int.

Nota

No se debe confundir las órdenes del historial de trading con las órdenes pendientes actuales que se muestran en la pestaña "Trading" del panel "Caja de Herramientas" del terminal de cliente. La lista de las órdenes que han sido canceladas o las que han llevado a la ejecución de una operación comercial se puede ver en la pestaña "Historial" del panel "Caja de Herramientas" del terminal de cliente.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- solicitamos toda la historia disponible en la cuenta
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- obtenemos el número de órdenes en la lista y mostramos el valor en el registro
   int total=HistoryOrdersTotal();
   Print("Number of historical orders on the account: ", total);
   /*
   resultado:
   Number of historical orders on the account: 496
   */
  }
```

Véase también

HistorySelect(), HistoryOrderSelect(), HistoryOrderGetTicket(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetTicket

Devuelve el ticket de una orden correspondiente en el historial. Antes de llamar a la función HistoryOrderGetTicket(), hay que recibir el historial de transacciones y órdenes usando la función HistorySelect() o HistorySelectByPosition().

```
ulong  HistoryOrderGetTicket(
   int  index      // número en la lista de órdenes
   );
```

Parámetros

index

[in] Número de la orden en la lista de órdenes.

Valor devuelto

Valor del tipo ulong. En caso de ejecución fallida devuelve 0.

Nota

No se debe confundir las órdenes del historial de trading con las órdenes pendientes actuales que se muestran en la pestaña "Trading" del panel "Caja de Herramientas" del terminal de cliente. La lista de las órdenes que han sido canceladas o las que han llevado a la ejecución de una operación comercial se puede ver en la pestaña "Historial" del panel "Caja de Herramientas" del terminal de cliente.

Ejemplo:

```
void OnStart()
  {
   datetime from=0;
   datetime to=TimeCurrent();
//--- solicitar todo el historial
   HistorySelect(from,to);
//--- variables para recibir valores desde las propiedades de la orden
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   datetime time_done;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- número de actuales órdenes pendientes
   uint     total=HistoryOrdersTotal();
//--- repasamos todas las órdenes en el ciclo
   for(uint i=0;i<total;i++)
     {
      //--- recibimos el ticket de la orden según su posición en la lista
      if((ticket=HistoryOrderGetTicket(i))>0)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- recibimos las propiedades de la orden
          open_price=       HistoryOrderGetDouble(ticket,ORDER_PRICE_OPEN);
          time_setup=       (datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_SETUP);
          time_done=        (datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_DONE);
          symbol=           HistoryOrderGetString(ticket,ORDER_SYMBOL);
          order_magic=      HistoryOrderGetInteger(ticket,ORDER_MAGIC);
          positionID =      HistoryOrderGetInteger(ticket,ORDER_POSITION_ID);
          initial_volume=   HistoryOrderGetDouble(ticket,ORDER_VOLUME_INITIAL);
          type=GetOrderType(HistoryOrderGetInteger(ticket,ORDER_TYPE));
          //--- preparamos y mostramos información sobre la orden
          printf("#ticket %d %s %G %s at %G was set up at %s => done at %s, pos ID=%d",
                 ticket,                   // ticket de la orden
                 type,                     // tipo
                 initial_volume,           // volumen colocado
                 symbol,                   // símbolo
                 open_price,               // precio de apertura especificado
                 TimeToString(time_setup),// hora de colocación de la orden
                 TimeToString(time_done), // hora de ejecución y eliminación
                 positionID                // ID de la posición en la que ha sido inclu
                 );
         }
      }
//---
   }
//+------------------------------------------------------------------+
//|  devuelve el nombre literal del tipo de la orden                 |
//+------------------------------------------------------------------+
string GetOrderType(long type)
   {
    string str_type="unknown operation";
    switch(type)
      {
       case (ORDER_TYPE_BUY):             return("buy");
       case (ORDER_TYPE_SELL):            return("sell");
       case (ORDER_TYPE_BUY_LIMIT):       return("buy limit");
       case (ORDER_TYPE_SELL_LIMIT):      return("sell limit");
       case (ORDER_TYPE_BUY_STOP):        return("buy stop");
       case (ORDER_TYPE_SELL_STOP):       return("sell stop");
       case (ORDER_TYPE_BUY_STOP_LIMIT): return("buy stop limit");
       case (ORDER_TYPE_SELL_STOP_LIMIT):return("sell stop limit");
      }
    return(str_type);
   }
```

Véase también

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetDouble

La función devuelve la propiedad de la orden que ha sido solicitada. La propiedad de la orden tiene que ser del tipo double. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
double  HistoryOrderGetDouble(
   ulong                      ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_DOUBLE  property_id        // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  HistoryOrderGetDouble(
   ulong                      ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,       // identificador de la propiedad
   double&                    double_var         // aquí recibimos el valor de la pro
   );
```

Parámetros

ticket_number

[in] Ticket de orden.

property_id

[in] Identificador de la propiedad de orden. Su valor puede ser uno de los valores de la enumeración ENUM_ORDER_PROPERTY_DOUBLE.

double_var

[out] Variable del tipo double que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo double.

Nota

No hay que confundir las órdenes del historial de trading con las órdenes pendientes actuales que se muestran en la pestaña "Trading" en el panel "Caja de Herramientas". La lista de las órdenes que han sido canceladas o han llevado a la ejecución de la operación comercial se puede encontrar en la pestaña "Historial" en el panel "Caja de Herramientas" del terminal de cliente.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- solicitamos la historia de transacciones y órdenes
   if(!HistorySelect(0, TimeCurrent()))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("HistorySelect() failed. Error ", GetLastError());
       return;
      }

//--- en un ciclo por la lista de todas las órdenes históricas en la cuenta
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el ticket de la orden en la lista según el índice del ciclo
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- obtenemos el tipo de la orden y mostramos el encabezado de la lista de pro
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket,
      PrintFormat("Double properties of an history order %s #%I64u:", type, ticket);

      //--- imprimimos bajo el encabezado todas las propiedades reales de la orden sel
      HistoryOrderPropertiesDoublePrint(ticket, 16);
     }
   /*
   resultado:
   Double properties of an history order Sell #2810847541:
   Volume initial: 0.50
   Volume current: 0.00
   Price open:     1.10491
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10491
   StopLimit:      0.00000
   Double properties of an history order Buy Limit #2811003507:
   Volume initial: 1.00
   Volume current: 1.00
   Price open:     1.10547
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10591
   StopLimit:      0.00000
   */
  }
//+------------------------------------------------------------------+
//| Muestra en el registro las propiedades reales                    |
//| de la orden histórica seleccionada                               |
//+------------------------------------------------------------------+
void HistoryOrderPropertiesDoublePrint(const long ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
```

© 2000-2025, MetaQuotes Ltd.

```
   double value=0;

//--- obtenemos el símbolo de la orden y el número de decimales del símbolo
   string symbol = HistoryOrderGetString(ticket, ORDER_SYMBOL);
   int    digits = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- definimos el texto del encabezado y la anchura del campo del encabezado
//--- si la anchura del encabezado se ha transmitido a la función como igual a cero, e
   header="Volume initial:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- obtenemos y mostramos en el registro el volumen inicial al colocar la orden con
   if(!HistoryOrderGetDouble(ticket, ORDER_VOLUME_INITIAL, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- mostramos en el registro el valor del volumen no ejecutado de la orden
   header="Volume current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_VOLUME_CURRENT, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- mostramos en el registro el valor del precio indicado en la orden
   header="Price open:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_OPEN, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- mostramos en el registro el valor del nivel de StopLoss
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- mostramos en el registro el valor del nivel de TakeProfit
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- mostramos en el registro el valor del precio actual según el símbolo de la orden
   header="Price current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_CURRENT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos en el registro el valor de precio de la colocación de la orden Limit a
   header="StopLimit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_STOPLIMIT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de orden                         |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type");
     }
  }
```

Véase también

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetInteger

La función devuelve la propiedad de la orden que ha sido solicitada. La propiedad de la orden tiene que ser del tipo datetime, int. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
long  HistoryOrderGetInteger(
   ulong                        ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  HistoryOrderGetInteger(
   ulong                        ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_INTEGER  property_id,       // identificador de la propiedad
   long&                        long_var           // aquí recibimos el valor de la pr
   );
```

Parámetros

ticket_number

[in] Ticket de orden.

property_id

[in] Identificador de la propiedad de orden. Su valor puede ser uno de los valores de la enumeración ENUM_ORDER_PROPERTY_INTEGER.

long_var

[out] Variable del tipo long que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo long.

Nota

No se debe confundir las órdenes del historial de trading con las órdenes pendientes actuales que se muestran en la pestaña "Trading" del panel "Caja de Herramientas" del terminal de cliente. La lista de las órdenes que han sido canceladas o las que han llevado a la ejecución de una operación comercial se puede ver en la pestaña "Historial" del panel "Caja de Herramientas" del terminal de cliente.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- obtenemos el ticket de la última orden desde el historial de trading semanal
   ulong last_order=GetLastOrderTicket();
   if(HistoryOrderSelect(last_order))
     {
      //--- tiempo de colocación de la orden en milisegundos desde el 01.01.1970
      long time_setup_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_SETUP_MSC);
      PrintFormat("Order #%d ORDER_TIME_SETUP_MSC=%i64 => %s",
                  last_order,time_setup_msc,TimeToString(time_setup_msc/1000));
      //--- tiempo de ejecución/eliminación de la orden en milisegundos desde el 01.01
      long  time_done_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_DONE_MSC);
      PrintFormat("Order #%d ORDER_TIME_DONE_MSC=%i64 => %s",
                  last_order,time_done_msc,TimeToString(time_done_msc/1000));
     }
   else // avisamos sobre el fracaso
      PrintFormat("HistoryOrderSelect() failed for #%d. Eror code=%d",
                  last_order,GetLastError());

//---
  }
//+------------------------------------------------------------------+
//|  devuelve el ticket de la última orden en el historial o -1      |
//+------------------------------------------------------------------+
ulong GetLastOrderTicket()
  {
//--- solicitamos el historial de lo últimos 7 días
   if(!GetTradeHistory(7))
     {
      //--- avisamos sobre la llamad fallida y devolveremos -1
      Print(__FUNCTION__," HistorySelect() ha devuelto false");
      return -1;
     }
//---
   ulong first_order,last_order,orders=HistoryOrdersTotal();
//--- si hay órdenes, empezamos a trabajar con ellas
   if(orders>0)
     {
      Print("Orders = ",orders);
      first_order=HistoryOrderGetTicket(0);
      PrintFormat("first_order = %d",first_order);
      if(orders>1)
        {
         last_order=HistoryOrderGetTicket((int)orders-1);
         PrintFormat("last_order = %d",last_order);
         return last_order;
        }
      return first_order;
     }
//--- no hemos encontrado ninguna orden, devolveremos -1
   return -1;
  }
//+------------------------------------------------------------------+
//| solicita el historial de los últimos días y devuelve false en caso de fallo
//+------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- fijamos un período de tiempo semanal para solicitar el historial de trading
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- hacemos la solicitud y comprobamos el resultado
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
      return false;
     }
//--- historial recibido con éxito
   return true;
  }
```

Véase también

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetString

La función devuelve la propiedad de la orden que ha sido solicitada. La propiedad de la orden tiene que ser del tipo string. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
string  HistoryOrderGetString(
   ulong                      ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_STRING  property_id       // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  HistoryOrderGetString(
   ulong                      ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_STRING  property_id,      // identificador de la propiedad
   string&                    string_var         // aquí recibimos el valor de la pro
   );
```

Parámetros

ticket_number

[in] Ticket de orden.

property_id

[in] Identificador de la propiedad de orden. Su valor puede ser uno de los valores de la enumeración ENUM_ORDER_PROPERTY_STRING.

string_var

[out] Variable del tipo string que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo string.

Nota

No se debe confundir las órdenes del historial de trading con las órdenes pendientes actuales que se muestran en la pestaña "Trading" del panel "Caja de Herramientas" del terminal de cliente. La lista de las órdenes que han sido canceladas o las que han llevado a la ejecución de una operación comercial se puede ver en la pestaña "Historial" del panel "Caja de Herramientas" del terminal de cliente.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- solicitamos la historia de transacciones y órdenes
   if(!HistorySelect(0, TimeCurrent()))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("HistorySelect() failed. Error ", GetLastError());
       return;
      }

//--- en un ciclo por la lista de todas las órdenes históricas en la cuenta
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos el ticket de la orden en la lista según el índice del ciclo
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- obtenemos el tipo de la orden y mostramos el encabezado de la lista de pro
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket,
      PrintFormat("String properties of an history order %s #%I64u:", type, ticket);

      //--- imprimimos bajo el encabezado todas las propiedades string de la orden sel
      HistoryOrderPropertiesStringPrint(ticket, 16);
     }
   /*
   resultado:
   String properties of an history order Buy #2646074112:
   Comment:         [tp 1.09137]
   Symbol:          EURUSD
   External ID:
   String properties of an history order Buy #2646131906:
   Comment:
   Symbol:          EURUSD
   External ID:
   */
  }
//+------------------------------------------------------------------+
//| Muestra en el registro las propiedades string                    |
//| de la orden histórica seleccionada                               |
//+------------------------------------------------------------------+
void HistoryOrderPropertiesStringPrint(const long ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
   string value="";

//--- definimos el texto del encabezado y la anchura del campo del encabezado
//--- si la anchura del encabezado se ha transmitido a la función como igual a cero, e
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- obtenemos y mostramos en el registro los comentarios con un encabezado de la anc
   if(!HistoryOrderGetString(ticket, ORDER_COMMENT, value))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- mostramos en el registro el símbolo del que se ha colocado la orden
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetString(ticket, ORDER_SYMBOL, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- mostramos en el registro el identificador de la orden en el sistema comercial ex
   header="External ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetString(ticket, ORDER_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de orden                         |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
```

Véase también

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propiedades de órdenes

© 2000-2025, MetaQuotes Ltd.

HistoryDealSelect

Elige en el historial una transacción para recurrir posteriormente a ésta a través de las funciones correspondientes. Devuelve true en caso de que la ejecución de la función se finalice con éxito, de lo contrario devuelve false. Para obtener la información sobre el error hay que llamar a la función GetLastError().

```
bool  HistoryDealSelect(
   ulong  ticket,     // ticket de la transacción
   );
```

Parámetros

ticket

[in] Ticket de la transacción.

Valor devuelto

Devuelve true en caso del éxito, de lo contrario devuelve false.

Nota

No se puede confundir las órdenes, transacciones y posiciones. Cada transacción es el resultado de la ejecución de una orden, mientras que cada posición es el resultado final de una o más transacciones.

La función HistoryDealSelect() limpia en un programa mql5 la lista de transacciones, las que están disponible, y copia en esta lista sólo una transacción, si la ejecución de la función HistoryDealSelect() se ha completado con éxito. Si hace falta repasar todas las transacciones seleccionadas por la función HistorySelect() es mejor utilizar la función HistoryDealGetTicket().

Ejemplo:

```
#define   TICKET    2620919264   // ticket de cualquier transacción conocida, por ejer

long      ExtTicket=TICKET;      // escribimos el ticket especificado en una variable
                                 // o bien procesamos las transacciones en el manejad
//+------------------------------------------------------------------+
//| Expert TradeTransaction handler                                  |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction& trans,
                        const MqlTradeRequest& request,
                        const MqlTradeResult& result)
  {
   //--- si es una transacción, añadimos la operación a la historia
   if(trans.type==TRADE_TRANSACTION_DEAL_ADD)
     {
      //--- seleccionamos una transacción según el ticket, obtenemos sus datos y mostr
      HistoryDealSelectProcess(trans.deal);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- seleccionamos una transacción según el ticket, obtenemos sus datos y mostramos l
   HistoryDealSelectProcess(ExtTicket);
   /*
   resultado:
   (Position ID #2645974677) EURUSD Deal Out 0.10 Buy #2620919264 by order #264602896
   */
  }
//+------------------------------------------------------------------+
//| Selecciona una operación según el ticket e imprime los datos de la misma en el reg
//+------------------------------------------------------------------+
void HistoryDealSelectProcess(const ulong deal_ticket)
  {
//--- seleccionamos una operación histórica según el ticket especificado en deal_ticke
   ResetLastError();
   if(!HistoryDealSelect(deal_ticket))
     {
      PrintFormat("HistoryDealSelect(%I64u) failed. Error %d", deal_ticket, GetLastErr
      return;
     }

//--- si la operación se ha seleccionado con éxito, obtenemos sus datos y mostramos la
   ENUM_DEAL_TYPE    deal_type  = (ENUM_DEAL_TYPE)HistoryDealGetInteger(ExtTicket, DEA
   ENUM_DEAL_ENTRY   deal_entry = (ENUM_DEAL_ENTRY)HistoryDealGetInteger(ExtTicket, DE
   ENUM_DEAL_REASON  deal_reason= (ENUM_DEAL_REASON)HistoryDealGetInteger(ExtTicket, I
   long              deal_time  = HistoryDealGetInteger(ExtTicket, DEAL_TIME_MSC);
   long              deal_order = HistoryDealGetInteger(ExtTicket, DEAL_ORDER);
   long              deal_pos_id= HistoryDealGetInteger(ExtTicket, DEAL_POSITION_ID);
   string            deal_symbol= HistoryDealGetString(ExtTicket, DEAL_SYMBOL);
   double            deal_volume= HistoryDealGetDouble(ExtTicket, DEAL_VOLUME);
   double            deal_price = HistoryDealGetDouble(ExtTicket, DEAL_PRICE);
   int               digits     = (int)SymbolInfoInteger(deal_symbol, SYMBOL_DIGITS);

   PrintFormat("(Position ID #%I64d) %s Deal %s %.2f %s #%I64u by order #%I64d at %.*f
               deal_pos_id, deal_symbol, DealEntryDescription(deal_entry), deal_volume
               DealTypeDescription(deal_type), ExtTicket, deal_order, digits, deal_pri
               TimeMscToString(deal_time));
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de transacción                   |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      case DEAL_TYPE_BUY                     :  return("Buy");
      case DEAL_TYPE_SELL                    :  return("Sell");
      case DEAL_TYPE_BALANCE                 :  return("Balance");
      case DEAL_TYPE_CREDIT                  :  return("Credit");
      case DEAL_TYPE_CHARGE                  :  return("Additional charge");
      case DEAL_TYPE_CORRECTION              :  return("Correction");
      case DEAL_TYPE_BONUS                   :  return("Bonus");
      case DEAL_TYPE_COMMISSION              :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY        :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY      :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY  :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY:  return("Monthly agent commission");
      case DEAL_TYPE_INTEREST                :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED            :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED           :  return("Canceled sell deal");
      case DEAL_DIVIDEND                     :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED             :  return("Franked (non-taxable) dividenc
      case DEAL_TAX                          :  return("Tax charges");
      default                                :  return("Unknown deal type: "+(string)t
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del método de cambio de posición          |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("Reverce");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
//+------------------------------------------------------------------+
//| Retorna la hora en milisegundos                                  |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SECO
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
  }
```

Véase también

HistorySelect(), HistoryDealGetTicket(), Propiedades de transacciones

© 2000-2025, MetaQuotes Ltd.

HistoryDealsTotal

Devuelve el número de transacciones en el historial. Antes de llamar a la función HistoryDealsTotal(), hay que recibir el historial de transacciones y órdenes usando la función HistorySelect() o HistorySelectByPosition().

```
int  HistoryDealsTotal();
```

Valor devuelto

Valor del tipo int.

Nota

No se puede confundir las órdenes, transacciones y posiciones. Cada transacción es el resultado de la ejecución de una orden, mientras que cada posición es el resultado final de una o más transacciones.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- solicitamos toda la historia disponible en la cuenta
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- obtenemos el número de operaciones y mostramos el valor en el registro
   int total=HistoryDealsTotal();
   Print("Number of historical deals on the account: ", total);
   /*
   resultado:
   Number of historical deals on the account: 339
   */
  }
```

Véase también

HistorySelect(), HistoryDealGetTicket(), Propiedades de transacciones

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetTicket

Elige una transacción a procesar posteriormente y devuelve el ticket de transacción en el historial. Antes de llamar a la función HistoryDealGetTicket(), hay que recibir el historial de transacciones y órdenes usando la función HistorySelect() o HistorySelectByPosition().

```
ulong  HistoryDealGetTicket(
   int  index      // número de transacción
   );
```

Parámetros

index

[in] Número de transacción en la lista de transacciones.

Valor devuelto

Valor del tipo ulong. En caso de ejecución fallida devuelve 0.

Nota

No se puede confundir las órdenes, transacciones y posiciones. Cada transacción es el resultado de la ejecución de una orden, mientras que cada posición es el resultado final de una o más transacciones.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   ulong deal_ticket;            // ticket de transacción
   ulong order_ticket;           // ticket de la orden según la cual ha sido realizada
   datetime transaction_time;    // hora de realizar la transacción
   long deal_type ;              // tipo de operación comercial
   long position_ID;             // identificador de la posición
   string deal_description;      // descripción de la operación
   double volume;                // volumen de operación
   string symbol;                // símbolo usado en la transacción
//--- vamos a fijar la fecha inicial y final para solicitar el historial de transacci
   datetime from_date=0;          // desde el principio
   datetime to_date=TimeCurrent();// hasta el momento actual
//--- vamos a solicitar el historial de transacciones del período especificado
   HistorySelect(from_date,to_date);
//--- número total en la lista de transacciones
   int deals=HistoryDealsTotal();
//--- ahora vamos a procesar cada transacción
   for(int i=0;i<deals;i++)
     {
      deal_ticket=               HistoryDealGetTicket(i);
      volume=                    HistoryDealGetDouble(deal_ticket,DEAL_VOLUME);
      transaction_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
      order_ticket=              HistoryDealGetInteger(deal_ticket,DEAL_ORDER);
      deal_type=                 HistoryDealGetInteger(deal_ticket,DEAL_TYPE);
      symbol=                    HistoryDealGetString(deal_ticket,DEAL_SYMBOL);
      position_ID=               HistoryDealGetInteger(deal_ticket,DEAL_POSITION_ID);
      deal_description=          GetDealDescription(deal_type,volume,symbol,order_tic
      //--- hagamos el formateado bonito para el número de transacción
      string print_index=StringFormat("% 3d",i);
      //--- mostramos la información sobre la transacción
      Print(print_index+": deal #",deal_ticket," at ",transaction_time,deal_descriptio
     }
  }
//+------------------------------------------------------------------+
//|  devuelve la descripción literal de la transacción               |
//+------------------------------------------------------------------+
string GetDealDescription(long deal_type,double volume,string symbol,long ticket,long
  {
   string descr;
//---
   switch(deal_type)
     {
      case DEAL_TYPE_BALANCE:                  return ("balance");
      case DEAL_TYPE_CREDIT:                   return ("credit");
      case DEAL_TYPE_CHARGE:                   return ("charge");
      case DEAL_TYPE_CORRECTION:               return ("correction");
      case DEAL_TYPE_BUY:                      descr="buy"; break;
      case DEAL_TYPE_SELL:                     descr="sell"; break;
      case DEAL_TYPE_BONUS:                    return ("bonus");
      case DEAL_TYPE_COMMISSION:               return ("additional commission");
      case DEAL_TYPE_COMMISSION_DAILY:         return ("daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY:       return ("monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY:   return ("daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return ("monthly agent commission");
      case DEAL_TYPE_INTEREST:                 return ("interest rate");
      case DEAL_TYPE_BUY_CANCELED:             descr="cancelled buy deal"; break;
      case DEAL_TYPE_SELL_CANCELED:            descr="cancelled sell deal"; break;
     }
   descr=StringFormat("%s %G %s (order #%d, position ID %d)",
                      descr,  // descripción corriente
```

© 2000-2025, MetaQuotes Ltd.

```
                        volume, // volumen de transacción
                        symbol, // instrumento de transacción
                        ticket, // ticket de la orden que ha provocado la transacción
                        pos_ID  // ID de la posición en la que ha participado la transac
                        );
   return(descr);
//---
  }
```

Véase también

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), Propiedades de transacciones

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetDouble

La función devuelve la propiedad solicitada de una transacción. La propiedad de la transacción tiene que ser del tipo double. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
double  HistoryDealGetDouble(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_DOUBLE  property_id        // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  HistoryDealGetDouble(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_DOUBLE  property_id,       // identificador de la propiedad
   double&                    double_var         // aquí recibimos el valor de la prop
   );
```

Parámetros

ticket_number

[in] Ticket de transacción.

property_id

[in] Identificador de la propiedad de transacción. Su valor puede ser uno de los valores de la enumeración ENUM_DEAL_PROPERTY_DOUBLE.

double_var

[out] Variable del tipo double que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo double.

Nota

No se puede confundir las órdenes, transacciones y posiciones. Cada transacción es el resultado de la ejecución de una orden, mientras que cada posición es el resultado final de una o más transacciones.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- solicitamos la historia de transacciones y órdenes
   if(!HistorySelect(0, TimeCurrent()))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- en un ciclo por la lista de transacciones en la historia de la cuenta
   int total=HistoryDealsTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos un ticket para la siguiente transacción (la transacción se selec
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;

      //--- obtenemos el tipo y la dirección de la transacción y mostramos el encabeza
      string type=DealTypeDescription((ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket, DE
      string entry=DealEntryDescription((ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket,
      PrintFormat("Double properties of an deal %s entry %s #%I64u:", type, entry, tic

      //--- imprimimos bajo el encabezado todas las propiedades reales de la transacci
      HistoryDealPropertiesDoublePrint(ticket, 12);
     }
   /*
   resultado:
   Double properties of an deal Buy entry In #2785070622:
   Volume:     0.50
   Price:      1.10480
   Commission: 0.00
   Swap:       0.00
   Profit:     0.00 USD
   Fee:        0.00
   StopLoss:   0.00000
   TakeProfit: 0.00000
   Double properties of an deal Sell entry Out #2785071538:
   Volume:     0.50
   Price:      1.10491
   Commission: 0.00
   Swap:       0.00
   Profit:     5.50 USD
   Fee:        0.00
   StopLoss:   0.00000
   TakeProfit: 0.00000
   */
  }
//+------------------------------------------------------------------+
//| Muestra en el registro las propiedades reales de la transacción elegida
//+------------------------------------------------------------------+
void HistoryDealPropertiesDoublePrint(const ulong ticket, const uint header_width=0)
  {
   uint   w=0;
```

© 2000-2025, MetaQuotes Ltd.

```
   string header="";
   double value=0;

//--- obtenemos el símbolo de la transacción, la divisa de beneficio y el número de de
   string symbol   = HistoryDealGetString(ticket, DEAL_SYMBOL);
   string currency= SymbolInfoString(symbol, SYMBOL_CURRENCY_PROFIT);
   int    digits   = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- definimos el texto del encabezado y la anchura del campo del encabezado
//--- si la anchura del encabezado se ha transmitido a la función como igual a cero, e
   header="Volume:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- obtenemos y mostramos en el registro el volumen de la transacción con un encabez
   if(!HistoryDealGetDouble(ticket, DEAL_VOLUME, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- mostramos en el registro el precio de la transacción
   header="Price:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_PRICE, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- mostramos en el registro la comisión por transacción
   header="Commission:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_COMMISSION, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- mostramos en el registro el swap acumulado en el cierre
   header="Swap:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_SWAP, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- mostramos en el registro el resultado financiero de la transacción
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_PROFIT, value))
      return;
   PrintFormat("%-*s%-.2f %s", w, header, value, currency);

//--- mostramos en el registro el pago por la realización de la transacción
   header="Fee:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_FEE, value))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- mostramos en el registro el valor del nivel de StopLoss
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- mostramos en el registro el valor del nivel de TakeProfit
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de transacción                   |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                    :  return("Buy");
      case DEAL_TYPE_SELL                   :  return("Sell");
      case DEAL_TYPE_BALANCE                :  return("Balance");
      case DEAL_TYPE_CREDIT                 :  return("Credit");
      case DEAL_TYPE_CHARGE                 :  return("Additional charge");
      case DEAL_TYPE_CORRECTION             :  return("Correction");
      case DEAL_TYPE_BONUS                  :  return("Bonus");
      case DEAL_TYPE_COMMISSION             :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY       :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY     :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return("Monthly agent commission");
      case DEAL_TYPE_INTEREST               :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED           :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED          :  return("Canceled sell deal");
      case DEAL_DIVIDEND                    :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED            :  return("Franked (non-taxable) dividend
      case DEAL_TAX                         :  return("Tax charges");
      default                               :  return("Unknown deal type: "+(string)t
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del método de cambio de posición          |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
```

© 2000-2025, MetaQuotes Ltd.

```
{
 switch(entry)
   {
    case DEAL_ENTRY_IN       :  return("In");
    case DEAL_ENTRY_OUT      :  return("Out");
    case DEAL_ENTRY_INOUT    :  return("Reverce");
    case DEAL_ENTRY_OUT_BY   :  return("Out by");
    case DEAL_ENTRY_STATE    :  return("Status record");
    default                  :  return("Unknown deal entry: "+(string)entry);
   }
}
```

Véase también

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Propiedades de transacciones

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetInteger

La función devuelve la propiedad solicitada de una transacción. La propiedad de la transacción tiene que ser del tipo datetime, int. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
long  HistoryDealGetInteger(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_INTEGER  property_id       // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  HistoryDealGetInteger(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_INTEGER  property_id,      // identificador de la propiedad
   long&                      long_var           // aquí recibimos el valor de la pro
   );
```

Parámetros

ticket_number

[in] Ticket de transacción.

property_id

[in] Identificador de la propiedad de transacción. Su valor puede ser uno de los valores de la enumeración ENUM_DEAL_PROPERTY_INTEGER.

long_var

[out] Variable del tipo long que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo long.

Nota

No se puede confundir las órdenes, transacciones y posiciones. Cada transacción es el resultado de la ejecución de una orden, mientras que cada posición es el resultado final de una o más transacciones.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- obtenemos el ticket de la última transacción desde el historial de trading semar
   ulong last_deal=GetLastDealTicket();
   if(HistoryDealSelect(last_deal))
     {
      //--- tiempo de ejecución de transacción en milisegundos desde 01.01.1970
      long deal_time_msc=HistoryDealGetInteger(last_deal,DEAL_TIME_MSC);
      PrintFormat("Deal #%d DEAL_TIME_MSC=%i64 => %s",
                  last_deal,deal_time_msc,TimeToString(deal_time_msc/1000));
     }
   else
      PrintFormat("HistoryDealSelect() failed for #%d. Eror code=%d",
                  last_deal,GetLastError());
//---
  }
//+------------------------------------------------------------------+
//|  devuelve el ticket de la última transacción en el historial o -1|
//+------------------------------------------------------------------+
ulong GetLastDealTicket()
  {
//--- solicitamos el historial de los últimos 7 días
   if(!GetTradeHistory(7))
     {
      //--- avisamos sobre la llamada fallida y devolvemos -1
      Print(__FUNCTION__," HistorySelect() ha devuelto false");
      return -1;
     }
//---
   ulong first_deal,last_deal,deals=HistoryDealsTotal();
//--- si hay órdenes, empezamos a trabajar con ellas
   if(deals>0)
     {
      Print("Deals = ",deals);
      first_deal=HistoryDealGetTicket(0);
      PrintFormat("first_deal = %d",first_deal);
      if(deals>1)
        {
         last_deal=HistoryDealGetTicket((int)deals-1);
         PrintFormat("last_deal = %d",last_deal);
         return last_deal;
        }
      return first_deal;
     }
//--- no hemos encontrado ninguna transacción, devolveremos -1
   return -1;
  }
//+------------------------------------------------------------------+
//| solicita el historial de los últimos días y devuelve false en caso de fallo
//+------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- fijamos un período de tiempo semanal para solicitar el historial de trading
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- hacemos la solicitud y comprobamos el resultado
   if(!HistorySelect(from,to))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
       return false;
      }
//--- historial recibido con éxito
   return true;
  }
```

Véase también

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Propiedades de transacciones

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetString

La función devuelve la propiedad solicitada de una transacción. La propiedad de la transacción tiene que ser del tipo string. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
string  HistoryDealGetString(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_STRING  property_id        // identificador de la propiedad
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de la propiedad se coloca en una variable receptora que el último parámetro pasa por referencia.

```
bool  HistoryDealGetString(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_STRING  property_id,       // identificador de la propiedad
   string&                    string_var         // aquí recibimos el valor de la prop
   );
```

Parámetros

ticket_number

[in] Ticket de transacción.

property_id

[in] Identificador de la propiedad de transacción. Su valor puede ser uno de los valores de la enumeración ENUM_DEAL_PROPERTY_STRING.

string_var

[out] Variable del tipo string que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo string.

Nota

No se puede confundir las órdenes, transacciones y posiciones. Cada transacción es el resultado de la ejecución de una orden, mientras que cada posición es el resultado final de una o más transacciones.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- solicitamos la historia de transacciones y órdenes
   if(!HistorySelect(0, TimeCurrent()))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- en un ciclo por la lista de transacciones en la historia de la cuenta
   int total=HistoryDealsTotal();
   for(int i=0; i<total; i++)
     {
      //--- obtenemos un ticket para la siguiente transacción (la transacción se selec
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;

      //--- obtenemos el tipo y la dirección de la transacción y mostramos el encabeza
      string type=DealTypeDescription((ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket, DE
      string entry=DealEntryDescription((ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket,
      PrintFormat("String properties of an deal %s entry %s #%I64u:", type, entry, tic

      //--- imprimimos bajo el encabezado todas las propiedades reales de la transacci
      HistoryDealPropertiesStringPrint(ticket, 13);
     }
   /*
   resultado:
   String properties of an deal Buy entry In #2785021084:
   Symbol:       EURUSD
   Comment:      Test PositionGetString
   Extarnal ID:
   String properties of an deal Buy entry Out #2497993663:
   Symbol:       EURUSD
   Comment:      [tp 1.08639]
   Extarnal ID:
   */
  }
//+------------------------------------------------------------------+
//| Muestra en el registro las propiedades string  de la transacción elegida
//+------------------------------------------------------------------+
void HistoryDealPropertiesStringPrint(const ulong ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
   string value ="";

//--- definimos el texto del encabezado y la anchura del campo del encabezado
//--- si la anchura del encabezado se ha transmitido a la función como igual a cero, e
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- obtenemos y mostramos en el registro el símbolo de la transacción con un encabez
   if(!HistoryDealGetString(ticket, DEAL_SYMBOL, value))
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("%-*s%-s", w, header, value);

//--- mostramos en el registro los comentarios a la transacción
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetString(ticket, DEAL_COMMENT, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- mostramos en el registro el identificador de la transacción en el sistema comerc
   header="Extarnal ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetString(ticket, DEAL_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del tipo de transacción                   |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                     :  return("Buy");
      case DEAL_TYPE_SELL                    :  return("Sell");
      case DEAL_TYPE_BALANCE                 :  return("Balance");
      case DEAL_TYPE_CREDIT                  :  return("Credit");
      case DEAL_TYPE_CHARGE                  :  return("Additional charge");
      case DEAL_TYPE_CORRECTION              :  return("Correction");
      case DEAL_TYPE_BONUS                   :  return("Bonus");
      case DEAL_TYPE_COMMISSION              :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY        :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY      :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY  :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY:  return("Monthly agent commission");
      case DEAL_TYPE_INTEREST                :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED            :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED           :  return("Canceled sell deal");
      case DEAL_DIVIDEND                     :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED             :  return("Franked (non-taxable) dividenc
      case DEAL_TAX                          :  return("Tax charges");
      default                                :  return("Unknown deal type: "+(string)t
     }
  }
//+------------------------------------------------------------------+
//| Retorna la descripción del método de cambio de posición          |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("Reverce");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
```

Véase también

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Propiedades de transacciones

© 2000-2025, MetaQuotes Ltd.

Administrar señales

Este grupo de funciones sirve para gestionar las señales comerciales. Estas funciones permiten lo siguiente:

- obtener la información sobre las señales disponibles para el copiado,
- ver o configurar los ajustes del copiado de las señales,
- suscribirse a la señal o darse de baja usando las funciones del lenguaje MQL5.

Función	Acción
SignalBaseGetDouble	Devuelve el valor de la propiedad del tipo double para la señal seleccionada
SignalBaseGetInteger	Devuelve el valor de la propiedad del tipo integer para la señal seleccionada
SignalBaseGetString	Devuelve el valor de la propiedad del tipo string para la señal seleccionada
SignalBaseSelect	Selecciona la señal de la base de datos de las señales disponibles en el terminal
SignalBaseTotal	Devuelve el número total de señales disponibles en el terminal
SignalInfoGetDouble	Devuelve el valor de la propiedad del tipo double de los ajustes del copiado de la señal
SignalInfoGetInteger	Devuelve el valor de la propiedad del tipo integer de los ajustes del copiado de la señal
SignalInfoGetString	Devuelve el valor de la propiedad del tipo string de los ajustes del copiado de la señal
SignalInfoSetDouble	Establece el valor de la propiedad del tipo double en los ajustes del copiado de la señal
SignalInfoSetInteger	Establece el valor de la propiedad del tipo integer en los ajustes del copiado de la señal
SignalSubscribe	Realiza la suscripción al copiado de la señal
SignalUnsubscribe	Da de baja la suscripción al copiado de la señal

© 2000-2025, MetaQuotes Ltd.

SignalBaseGetDouble

Devuelve el valor de la propiedad del tipo double para la señal seleccionada.

```
double  SignalBaseGetDouble(
   ENUM_SIGNAL_BASE_DOUBLE     property_id,     // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de la señal. El valor puede ser uno de los valores de la enumeración ENUM_SIGNAL_BASE_DOUBLE.

Valor devuelto

Valor del tipo double de la propiedad de la señal especificada.

© 2000-2025, MetaQuotes Ltd.

SignalBaseGetInteger

Devuelve el valor de la propiedad del tipo integer para la señal seleccionada.

```
long  SignalBaseGetInteger(
   ENUM_SIGNAL_BASE_INTEGER     property_id,     // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de la señal. El valor puede ser uno de los valores de la enumeración ENUM_SIGNAL_BASE_INTEGER.

Valor devuelto

Valor del tipo integer de la propiedad de la señal especificada.

© 2000-2025, MetaQuotes Ltd.

SignalBaseGetString

Devuelve el valor de la propiedad del tipo string para la señal seleccionada.

```
string  SignalBaseGetString(
   ENUM_SIGNAL_BASE_STRING     property_id,     // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de la señal. El valor puede ser uno de los valores de la enumeración ENUM_SIGNAL_BASE_STRING.

Valor devuelto

Valor del tipo string de la propiedad de la señal especificada.

© 2000-2025, MetaQuotes Ltd.

SignalBaseSelect

Selecciona la señal de la base de datos de las señales disponibles en el terminal.

```
bool  SignalBaseSelect(
   int      index     // índice de la señal
   );
```

Parámetros

index

[in] Índice de la señal en la base de las señales comerciales.

Valor devuelto

Devuelve true si la función se completa con éxito, en caso del error, false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Ejemplo:

```
void OnStart()
  {
//--- solicita el número total de las señales en la base de datos
   int total=SignalBaseTotal();
//--- ciclo por todas las señales
   for(int i=0;i<total;i++)
     {
      //--- seleccionamos la señal para el trabajo posterior
      if(SignalBaseSelect(i))
        {
         //--- obtenemos las propiedades de la señal
         long   id    =SignalBaseGetInteger(SIGNAL_BASE_ID);          // id de la seña
         long   pips  =SignalBaseGetInteger(SIGNAL_BASE_PIPS);        // resultado del
         long   subscr=SignalBaseGetInteger(SIGNAL_BASE_SUBSCRIBERS); // número de sus
         string name  =SignalBaseGetString(SIGNAL_BASE_NAME);         // nombre de la
         double price =SignalBaseGetDouble(SIGNAL_BASE_PRICE);        // precio de sus
         string curr  =SignalBaseGetString(SIGNAL_BASE_CURRENCY);     // divisa de la
         //--- mostramos todas las señales rentables gratuitas con el número de suscri
         if(price==0.0 && pips>0 && subscr>0)
            PrintFormat("id=%d, name=\"%s\", currency=%s, pips=%d, subscribers=%d",id,
        }
      else PrintFormat("Error al seleccionar la señal. Código del error=%d",GetLastErr
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SignalBaseTotal

Devuelve el número total de las señales disponibles en el terminal.

```
int  SignalBaseTotal();
```

Valor devuelto

El número total de las señales disponibles en el terminal.

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetDouble

Devuelve el valor de la propiedad del tipo double de los ajustes del copiado de la señal.

```
double  SignalInfoGetDouble(
   ENUM_SIGNAL_INFO_DOUBLE     property_id,     // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de los ajustes del copiado de la señal. El valor puede ser uno de los valores de la enumeración ENUM_SIGNAL_INFO_DOUBLE.

Valor devuelto

Valor del tipo double de la propiedad especificada de los ajustes del copiado de la señal.

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetInteger

Devuelve el valor de la propiedad del tipo integer de los ajustes del copiado de la señal.

```
long  SignalInfoGetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de los ajustes del copiado de la señal. El valor puede ser uno de los valores de la enumeración ENUM_SIGNAL_INFO_INTEGER.

Valor devuelto

Valor del tipo integer de la propiedad especificada de los ajustes del copiado de la señal.

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetString

Devuelve el valor de la propiedad del tipo string de los ajustes del copiado de la señal.

```
string  SignalInfoGetString(
   ENUM_SIGNAL_INFO_STRING     property_id,     // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de los ajustes del copiado de la señal. El valor puede ser uno de los valores de la enumeración ENUM_SIGNAL_INFO_STRING.

Valor devuelto

Valor del tipo string de la propiedad especificada de los ajustes del copiado de la señal.

© 2000-2025, MetaQuotes Ltd.

SignalInfoSetDouble

Establece el valor de la propiedad del tipo double en los ajustes del copiado de la señal.

```
bool  SignalInfoSetDouble(
   ENUM_SIGNAL_INFO_DOUBLE      property_id,     // identificador de la propiedad
   double                       value            // valor de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de los ajustes del copiado de la señal. El valor puede ser uno de los valores de la enumeración ENUM_SIGNAL_INFO_DOUBLE.

value

[in] Valor de la propiedad de los ajustes del copiado de la señal.

Valor devuelto

Devuelve true si la propiedad se ha cambiado con éxito, de lo contrario devuelve false. Para obtener la información adicional sobre el error, hay que llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

SignalInfoSetInteger

Establece el valor de la propiedad del tipo integer en los ajustes del copiado de la señal.

```
bool  SignalInfoSetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // identificador de la propiedad
   long                         value            // valor de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad de los ajustes del copiado de la señal. El valor puede ser uno de los valores de la enumeración ENUM_SIGNAL_INFO_INTEGER.

value

[in] Valor de la propiedad de los ajustes del copiado de la señal.

Valor devuelto

Devuelve true si la propiedad se ha cambiado con éxito, de lo contrario devuelve false. Para obtener la información adicional sobre el error, hay que llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

SignalSubscribe

Realiza la suscripción al copiado de la señal especificada.

```
bool  SignalSubscribe(
   long     signal_id     // id de la señal
   );
```

Parámetros

signal_id

[in] Identificador de la señal.

Valor devuelto

Devuelve true si la suscripción al copiado de la señal se ha completado con éxito, de lo contrario devuelve false. Para obtener la información adicional sobre el error, hay que llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

SignalUnsubscribe

Da de baja la suscripción al copiado de la señal.

```
bool  SignalUnsubscribe();
```

Valor devuelto

Devuelve true si la suscripción al copiado de la señal se ha dado de baja con éxito, de lo contrario devuelve false. Para obtener la información adicional sobre el error, hay que llamar a la función GetLastError().

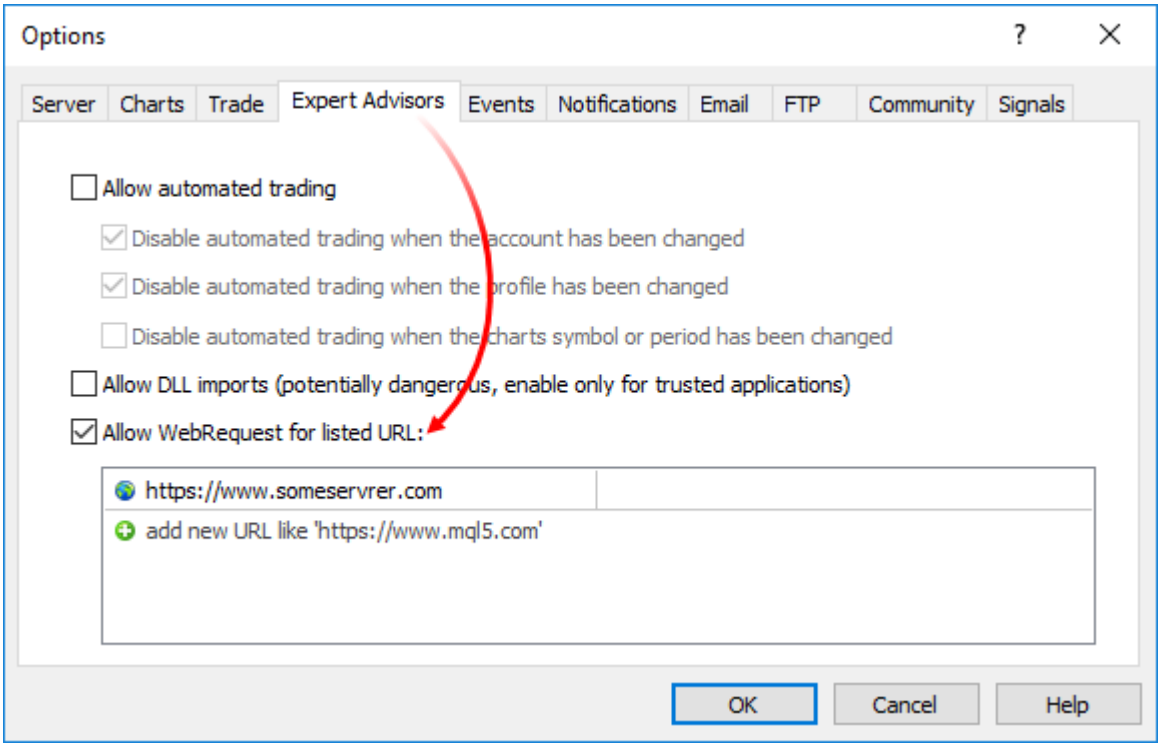
© 2000-2025, MetaQuotes Ltd.

Funciones de red

Los programas MQL5 pueden intercambiar datos con servidores remotos, enviar notificaciones push, correos electrónicos y datos por FTP.

- El grupo de funciones Socket* permite crear una conexión TCP (incluyendo una conexión TLS protegida) con un host remoto a través de sockets de sistema. El esquema de trabajo es sencillo: usted crea un socket, se conecta al servidor y puede comenzar la lectura y registro de datos.
- La función WebRequest ha sido pensada para trabajar con recursos web y permite enviar fácilmente solicitudes HTTP (incluyendo GET y POST).
- SendFTP, SendMail y SendNotification son las funciones más sencillas para enviar archivos, correos electrónicos y notificaciones móviles.

Para mayor seguridad, el usuario final en el lado del terminal de cliente se ha implementado una lista de direcciones IP permitidas, con las que se puede conectar un programa MQL5 con las ayuda de las funciones Socket* y WebRequest. Por ejemplo, si requiere la conexión https://www.someserver.com, esta deberá ser indicada de forma explícita por el usuario del terminal en la lista de permitidas. No es posible añadir la dirección de forma programática.



Para notificar al usuario sobre la necesidad de realizar ajustes adicionales, añada al programa MQL5 un mensaje explícito. Por ejemplo, a través de #property description, Alert o Print.

Función	Acción
SocketCreate	Crea un socket con las banderas indicadas y retorna su manejador
SocketClose	Cierra el socket
SocketConnect	Ejecuta la conexión al servidor, con control de límite de tiempo
SocketIsConnected	Comprueba si está conectado el socket en el momento actual

© 2000-2025, MetaQuotes Ltd.

Función	Acción
SocketIsReadable	Obtiene el número de bytes que se pueden leer desde el socket
SocketIsWritable	Comprueba si es posible registrar datos en el socket en el momento actual
SocketTimeouts	Establece el límite de tiempo para obtener y enviar los datos para el objeto de sistema del socket
SocketRead	Lee los datos desde el socket
SocketSend	Registra los datos en el socket
SocketTlsHandshake	Inicia una conexión TLS (SSL) protegida con el host indicado según el protocolo TLS Handshake
SocketTlsCertificate	Obtiene los datos sobre el certificado usado para proteger la conexión de red
SocketTlsRead	Lee los datos de una conexión TLS protegida
SocketTlsReadAvailable	Lee todos los datos disponibles de una conexión TLS protegida
SocketTlsSend	Envía los datos a través de una conexión TLS protegida
WebRequest	Envía una solicitud HTTP al servidor indicado
SendFTP	Envía un archivo según la dirección indicada en la ventana de ajustes en la pestaña "FTP"
SendMail	Envía un correo electrónico según la dirección indicada en la ventana de ajustes en la pestaña "Correo"
SendNotification	Envía una notificación Push a los terminales móviles cuyos MetaQuotes ID se han indicado en la pestaña "Notificaciones"

© 2000-2025, MetaQuotes Ltd.

SocketCreate

Crea un socket con las banderas indicadas y retorna su manejador.

```
int  SocketCreate(
   uint     flags          // banderas
   );
```

Parámetros

flags

[in] Combinación de banderas que determina el modo de trabajo con el socket. En este momento, se da soporte a una bandera — SOCKET_DEFAULT.

Valor retornado

Si el socket se crea con éxito, retorna su manejador, de lo contrario, INVALID_HANDLE.

Observaciones

Para liberar la memoria de la computadora de un socket no utilizado, llame para él SocketClose.

De un programa MQL5 se pueden crear un máximo de 128 sockets. Si se supera el límite, en _LastError se registrará el error 5271 (ERR_NETSOCKET_TOO_MANY_OPENED).

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para que el ejemplo funcione, añada Address a la lista de permi
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos al servidor                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- si se usa una conexión TLS protegida a través del puerto 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si se usa una conexión TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- leemos los datos del socket mientras haya, pero no por un tiempo superior a time
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de lectura, dependiendo de si la conexión está o no
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analizamos la respuesta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Obtenido el encabezado HTTP de la respuesta:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   int socket=SocketCreate();
//--- comprobando el manejador
   if(socket!=INVALID_HANDLE)
     {
      //--- si todo está en orden, nos conectamos
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conexión establecida con ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la conexión está protegida con un certificado, mostramos sus datos
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Propietario:  ",subject);
            Print("   Emisor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Huella digital: ",thumbprint);
            Print("   Expiración: ",expiration);
            ExtTLS=true;
           }
         //--- enviamos al servidor una solicitud GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitud GET enviada");
            //--- leemos la respuesta
            if(!HTTPRecv(socket,1000))
               Print("No se ha podido obtener la respuesta, error ",GetLastError());
           }
         else
            Print("No se ha logrado enviar la solicitud GET, error ",GetLastError());
        }
      else
        {
         Print("No se ha logrado conectar con ",Address,":",Port," error ",GetLastErro
        }
      //--- cerramos el socket después de usarlo
      SocketClose(socket);
     }
   else
      Print("No se ha logrado crear el socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketClose

Cierra el socket.

```
bool  SocketClose(
   const int  socket      // manejador del socket
   );
```

Parámetros

socket

[in] Manejador del socket que se debe cerrar. El manejador es retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Valor retornado

Retorna true en caso de éxito, de lo contrario, false.

Observación

Si para el socket se ha creado anteriormente una conexión a través de SocketConnect, esta será interrumpida.

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                  SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para que el ejemplo funcione, añada Address a la lista de permi
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos al servidor                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- si se usa una conexión TLS protegida a través del puerto 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si se usa una conexión TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- leemos los datos del socket mientras haya, pero no por un tiempo superior a time
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de lectura, dependiendo de si la conexión está o no
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analizamos la respuesta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Obtenido el encabezado HTTP de la respuesta:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   int socket=SocketCreate();
//--- comprobando el manejador
   if(socket!=INVALID_HANDLE)
     {
      //--- si todo está en orden, nos conectamos
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conexión establecida con ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la conexión está protegida con un certificado, mostramos sus datos
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Propietario:  ",subject);
            Print("   Emisor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Huella digital: ",thumbprint);
            Print("   Expiración: ",expiration);
            ExtTLS=true;
           }
         //--- enviamos al servidor una solicitud GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitud GET enviada");
            //--- leemos la respuesta
            if(!HTTPRecv(socket,1000))
               Print("No se ha podido obtener la respuesta, error ",GetLastError());
           }
         else
            Print("No se ha logrado enviar la solicitud GET, error ",GetLastError());
        }
      else
        {
         Print("No se ha logrado conectar con ",Address,":",Port," error ",GetLastErro
        }
      //--- cerramos el socket después de usarlo
      SocketClose(socket);
     }
   else
      Print("No se ha logrado crear el socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketConnect

Ejecuta la conexión al servidor, con control de límite de tiempo.

```
bool  SocketConnect(
   int           socket,              // socket
   const string  server,              // dirección para la conexión
   uint          port,                // puerto para la conexión
   uint          timeout_receive_ms   // tiempo límite de conexión
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

server

[in] Nombre de dominio del servidor al que debemos conectarnos, o su dirección IP.

port

[in] Número de puerto para la conexión.

timeout_receive_ms

[in] Límite de tiempo de la conexión en milisegundos. Si durante este tiempo no se logra establecer la conexión, los intentos se interrumpirán.

Valor retornado

Si la conexión tiene éxito, retornará true, de lo contrario, false.

Observación

La dirección para la conexión debe añadirse a la lista de permitidas en lado del terminal de cliente (apartado Servicio \ Ajustes \ Asesores).

Si se da un error de conexión a _LastError, se registrará el error 5272 (ERR_NETSOCKET_CANNOT_CONNECT).

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para que el ejemplo funcione, añada Address a la lista de permi
```

© 2000-2025, MetaQuotes Ltd.

```
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos al servidor                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si se usa una conexión TLS protegida a través del puerto 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si se usa una conexión TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- leemos los datos del socket mientras haya, pero no por un tiempo superior a time
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de lectura, dependiendo de si la conexión está o no
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analizamos la respuesta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Obtenido el encabezado HTTP de la respuesta:");
```

© 2000-2025, MetaQuotes Ltd.

```
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- comprobando el manejador
   if(socket!=INVALID_HANDLE)
     {
      //--- si todo está en orden, nos conectamos
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conexión establecida con ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la conexión está protegida con un certificado, mostramos sus datos
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Propietario:  ",subject);
            Print("   Emisor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Huella digital: ",thumbprint);
            Print("   Expiración: ",expiration);
            ExtTLS=true;
           }
         //--- enviamos al servidor una solicitud GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitud GET enviada");
            //--- leemos la respuesta
            if(!HTTPRecv(socket,1000))
               Print("No se ha podido obtener la respuesta, error ",GetLastError());
           }
         else
            Print("No se ha logrado enviar la solicitud GET, error ",GetLastError());
        }
      else
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         Print("No se ha logrado conectar con ",Address,":",Port," error ",GetLastErro
        }
      //--- cerramos el socket después de usarlo
      SocketClose(socket);
     }
   else
      Print("No se ha logrado crear el socket, error ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

SocketIsConnected

Comprueba si está conectado el socket en el momento actual.

```
bool  SocketIsConnected(
   const int  socket      // manejador del socket
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate(). Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Valor retornado

Retorna true si el socket está conectado, de lo contrario, false.

Observación

Con la ayuda de la función SocketIsConnected(), es posible comprobar la conexión del socket en este momento.

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para que el ejemplo funcione, añada Address a la lista de permi
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comando al servidor                                     |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si se utiliza una conexión protegida por TLS a través del puerto 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si se utiliza una conexión TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- leemos los datos del socket mientras esté disponible, pero sin superar el timeou
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de lectura dependiendo de si la conexión es segura
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- разберем ответ
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Encabezado de respuesta HTTP recibido:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   int socket=SocketCreate();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- si todo está en orden, nos conectamos
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Establecida conexión a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la conexión está protegida por un certificado, mostraremos sus datos
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Propietario:  ",subject);
            Print("   Emisor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Huella: ",thumbprint);
            Print("   Expiración: ",expiration);
            ExtTLS=true;
           }
         //--- enviamos al servidor una solicitud GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitud GET enviada");
            //--- leemos la respuesta
            if(!HTTPRecv(socket,1000))
               Print("No se ha podido obtener una respuesta, error ",GetLastError());
           }
         else
            Print("No se ha podido enviar la solicitud GET, error ",GetLastError());
        }
      else
        {
         Print("La conexión a ",Address,":",Port," ha fallado, error ",GetLastError())
        }
      //--- cerramos el socket después del uso
      SocketClose(socket);
     }
   else
      Print("No se ha podido crear el socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Mire también

SocketConnect, SocketIsWritable, SocketCreate, SocketClose

© 2000-2025, MetaQuotes Ltd.

SocketIsReadable

Obtiene el número de bytes que se pueden leer desde el socket.

```
uint  SocketIsReadable(
   const int  socket      // manejador del socket
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Valor retornado

Número de bytes que se pueden leer. En caso de error, retorna 0.

Observación

Si al ejecutar esta función aparece un error en el socket de sistema, la conexión establecida a través de SocketConnect será interrumpida.

Antes de llamar SocketRead, compruebe si hay en el socket datos para la lectura. En caso contrario, si no hay datos, la función SocketRead esperará en vano los datos durante timeout_ms, retrasando la ejecución del programa.

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para que el ejemplo funcione, añada Address a la lista de permi
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos al servidor                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
```

© 2000-2025, MetaQuotes Ltd.

```
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si se usa una conexión TLS protegida a través del puerto 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si se usa una conexión TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- leemos los datos del socket mientras haya, pero no por un tiempo superior a time
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de lectura, dependiendo de si la conexión está o no
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analizamos la respuesta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Obtenido el encabezado HTTP de la respuesta:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- comprobando el manejador
   if(socket!=INVALID_HANDLE)
     {
      //--- si todo está en orden, nos conectamos
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conexión establecida con ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la conexión está protegida con un certificado, mostramos sus datos
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Propietario:  ",subject);
            Print("   Emisor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Huella digital: ",thumbprint);
            Print("   Expiración: ",expiration);
            ExtTLS=true;
           }
         //--- enviamos al servidor una solicitud GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitud GET enviada");
            //--- leemos la respuesta
            if(!HTTPRecv(socket,1000))
               Print("No se ha podido obtener la respuesta, error ",GetLastError());
           }
         else
            Print("No se ha logrado enviar la solicitud GET, error ",GetLastError());
        }
      else
        {
         Print("No se ha logrado conectar con ",Address,":",Port," error ",GetLastErro
        }
      //--- cerramos el socket después de usarlo
      SocketClose(socket);
     }
   else
      Print("No se ha logrado crear el socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketIsWritable

Comprueba si es posible registrar datos en el socket en el momento actual.

```
bool  SocketIsWritable(
   const int  socket      // manejador del socket
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Valor retornado

Retorna true si el registro es posible, de lo contrario,false.

Observación

Al usar esta función, usted podrá comprobar si es posible registrar datos en el socket ahora mismo.

Si al ejecutar esta función aparece un error en el socket de sistema, la conexión establecida a través de SocketConnect será interrumpida.

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                              SocketIsWritable.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal setting
#property script_show_inputs

input string Address    ="www.mql5.com";
input int    Port       =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- creamos un socket y obtenemos su handle
   int socket=SocketCreate();
//--- проверим хэндл
```

© 2000-2025, MetaQuotes Ltd.

```
if(socket!=INVALID_HANDLE)
  {
   //--- si todo está en orden, nos conectamos
   if(SocketConnect(socket,Address,Port,1000))
     {
      PrintFormat("Established connection to %s:%d",Address,Port);

      string   subject,issuer,serial,thumbprint;
      datetime expiration;
      //--- si la conexión está protegida por un certificado, mostraremos sus datos
      if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
        {
         Print("TLS certificate:");
         Print("   Owner:       ",subject);
         Print("   Issuer:      ",issuer);
         Print("   Number:      ",serial);
         Print("   Print:       ",thumbprint);
         Print("   Expiration: ",expiration);
         ExtTLS=true;
        }
      //--- enviamos al servidor una solicitud GET
      if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
        {
         Print("GET request sent");
         //--- leemos la respuesta
         if(!HTTPRecv(socket,1000))
            Print("Failed to get a response, error ",GetLastError());
        }
      else
         Print("Failed to send GET request, error ",GetLastError());
     }
   else
     {
      PrintFormat("Connection to %s:%d failed, error %d",Address,Port,GetLastError
     }
   //--- imprimimos en el registro la posibilidad de escribir datos en el socket en
   if(SocketIsWritable(socket))
      Print("At the current moment in time, writing data to the socket is possible"
   else
      Print("It is not possible to write data to the socket at the current time");
   //--- cerramos el socket después del uso
   if(SocketClose(socket))
      Print("Now the socket is closed");
  }
else
   Print("Failed to create a socket, error ",GetLastError());
/*
resultado:
At the current moment in time, writing data to the socket is possible
```

© 2000-2025, MetaQuotes Ltd.

```
   Socket is closed now
   */
  }
//+------------------------------------------------------------------+
//| Enviando comando al servidor                                     |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
//--- convertimos la cadena en un array de caracteres, el cero final se descartará
   char req[];
   int  len=StringToCharArray(request,req)-1;

   if(len<0)
      return(false);
//--- si se utiliza una conexión protegida por TLS a través del puerto 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si se utiliza una conexión TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
  {
   char   rsp[];
   string result;
   ulong  timeout_check=GetTickCount64()+timeout_ms;
//--- leemos los datos del socket mientras esté disponible, pero sin superar el timeou
   do
     {
      uint len=SocketIsReadable(socket);

      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de lectura dependiendo de si la conexión es segura
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout_ms);
         //--- разберем ответ
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
            int header_end=StringFind(result,"\r\n\r\n");

            if(header_end>0)
```

© 2000-2025, MetaQuotes Ltd.

```
               {
                Print("HTTP answer header received:");
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
             //--- actualizamos el tiempo de timeout de la lectura
             timeout_check=GetTickCount64()+timeout_ms;
            }
         }
      }
   while(GetTickCount64()<timeout_check && !IsStopped());

   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

SocketTimeouts

Establece el límite de tiempo para obtener y enviar los datos para el objeto de sistema del socket.

```
bool  SocketTimeouts(
   int           socket,              // socket
   uint          timeout_send_ms,     // límite de tiempo de envío de datos
   uint          timeout_receive_ms   // límite de tiempo de obtención de datos
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

timeout_send_ms

[in] Límite de tiempo de envío de datos en milisegundos.

timeout_receive_ms

[in] Límite de tiempo de obtención de datos en milisegundos.

Valor retornado

Retorna true en caso de éxito, de lo contrario, false.

Observación

No confunda los límites de tiempo de los objetos de sistema con los límites de tiempo establecidos al leer datos a través de SocketRead. SocketTimeout establece los límites de tiempo una vez para el objeto del socket en el sistema operativo. Estos límites de tiempo se aplican a todas las funciones de lectura y envío de datos a través de este socket. En SocketRead, el límite de tiempo se establece para una operación concreta de lectura de datos.

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketTimeouts.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version     "1.00"

#define   TIMEOUT_SEND     5000
#define   TIMEOUT_RECEIVE  5000
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- creamos un socket y obtenemos su handle
   int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- establecemos los timeout para recibir y enviar datos para el objeto de sistema c
   if(SocketTimeouts(socket,TIMEOUT_SEND,TIMEOUT_RECEIVE))
      PrintFormat("timeouts were successfully set",socket,TIMEOUT_SEND,TIMEOUT_RECEIVE
   else
      PrintFormat("SocketTimeouts(%d, %I64d, %I64d) failed. Error %d",socket,TIMEOUT_S
//--- cerramos el socket
   SocketClose(socket);
  }
```

© 2000-2025, MetaQuotes Ltd.

SocketRead

Lee los datos desde el socket.

```
int  SocketRead(
   int           socket,              // socket
   uchar&        buffer[],            // búfer para leer los datos del socket
   uint          buffer_maxlen,       // número de bytes que deben ser leídos
   uint          timeout_ms           // límite de tiempo de lectura
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Enlace a la matriz de tipo uchar en la que se leerán los datos. El tamaño de la matriz dinámica aumenta en el número de bytes leídos. El tamaño de la matriz no puede superar INT_MAX (2147483647).

buffer_maxlen

[in] Número de bytes que deben ser leídos en la matriz *buffer[]*. Los datos que no han cabido en la matriz se quedarán en el socket. Será posible obtenerlos con la siguiente llamada de SocketRead. El valor *buffer_maxlen* no puede superar INT_MAX (2147483647).

timeout_ms

[in] Límite de tiempo de lectura de datos en milisegundos. Si durante este tiempo no logramos obtener los datos, los intentos finalizarán y la función retornará -1.

Valor retornado

En caso de éxito, retorna el número de bytes leídos, en caso de error, retorna -1.

Observación

Si al ejecutar esta función aparece un error en el socket de sistema, la conexión establecida a través de SocketConnect será interrumpida.

Si se da un error de lectura de datos, en _LastError se registrará el error 5273 (ERR_NETSOCKET_IO_ERROR).

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para que el ejemplo funcione, añada Address a la lista de permi
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos al servidor                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si se usa una conexión TLS protegida a través del puerto 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si se usa una conexión TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- leemos los datos del socket mientras haya, pero no por un tiempo superior a time
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de lectura, dependiendo de si la conexión está o no
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analizamos la respuesta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
```

© 2000-2025, MetaQuotes Ltd.

```
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Obtenido el encabezado HTTP de la respuesta:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
            }
          }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- comprobando el manejador
   if(socket!=INVALID_HANDLE)
     {
      //--- si todo está en orden, nos conectamos
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conexión establecida con ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la conexión está protegida con un certificado, mostramos sus datos
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Propietario:  ",subject);
            Print("   Emisor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Huella digital: ",thumbprint);
            Print("   Expiración: ",expiration);
            ExtTLS=true;
           }
         //--- enviamos al servidor una solicitud GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitud GET enviada");
            //--- leemos la respuesta
            if(!HTTPRecv(socket,1000))
               Print("No se ha podido obtener la respuesta, error ",GetLastError());
           }
         else
```

© 2000-2025, MetaQuotes Ltd.

```
            Print("No se ha logrado enviar la solicitud GET, error ",GetLastError());
         }
       else
         {
          Print("No se ha logrado conectar con ",Address,":",Port," error ",GetLastErro
         }
       //--- cerramos el socket después de usarlo
       SocketClose(socket);
      }
    else
       Print("No se ha logrado crear el socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Mire también

SocketTimeouts, MathSwap

© 2000-2025, MetaQuotes Ltd.

SocketSend

Registra los datos en el socket.

```
int  SocketSend(
   int           socket,              // socket
   const uchar&  buffer[],            // búfer para los datos
   uint          buffer_len           // tamaño del búfer
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[in] Enlace a la matriz de tipo uchar con los datos que deben ser enviados al socket.

buffer_len

[in] Tamaño de la matriz buffer.

Valor retornado

En caso de éxito, retorna el número de bytes registrados en el socket. En caso de error, retorna -1.

Observación

Si al ejecutar esta función aparece un error en el socket de sistema, la conexión establecida a través de SocketConnect será interrumpida.

Si se da un error de registro de datos, en _LastError se registrará el error 5273 (ERR_NETSOCKET_IO_ERROR).

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para que el ejemplo funcione, añada Address a la lista de permi
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
```

© 2000-2025, MetaQuotes Ltd.

```
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos al servidor                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si se usa una conexión TLS protegida a través del puerto 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si se usa una conexión TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- leemos los datos del socket mientras haya, pero no por un tiempo superior a time
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de lectura, dependiendo de si la conexión está o no
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analizamos la respuesta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Obtenido el encabezado HTTP de la respuesta:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      }
    while(GetTickCount()<timeout_check && !IsStopped());
    return(false);
   }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- comprobando el manejador
   if(socket!=INVALID_HANDLE)
     {
      //--- si todo está en orden, nos conectamos
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conexión establecida con ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la conexión está protegida con un certificado, mostramos sus datos
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Propietario:  ",subject);
            Print("   Emisor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Huella digital: ",thumbprint);
            Print("   Expiración: ",expiration);
            ExtTLS=true;
           }
         //--- enviamos al servidor una solicitud GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitud GET enviada");
            //--- leemos la respuesta
            if(!HTTPRecv(socket,1000))
               Print("No se ha podido obtener la respuesta, error ",GetLastError());
           }
         else
            Print("No se ha logrado enviar la solicitud GET, error ",GetLastError());
        }
      else
        {
         Print("No se ha logrado conectar con ",Address,":",Port," error ",GetLastErro
        }
      //--- cerramos el socket después de usarlo
      SocketClose(socket);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   else
      Print("No se ha logrado crear el socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Mire también

SocketTimeouts, MathSwap, StringToCharArray

© 2000-2025, MetaQuotes Ltd.

SocketTlsHandshake

Inicia una conexión TLS (SSL) protegida con el host indicado según el protocolo TLS Handshake. Durante el Handshake, el cliente y el servidor coordinan los parámetros de conexión: la versión del protocolo utilizado y el método de cifrado de los datos.

```
bool  SocketTlsHandshake(
   int           socket,               // socket
   const string  host                  // dirección del host
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

host

[in] Dirección del host con el que se establece la conexión protegida.

Valor retornado

Retorna true en caso de éxito, de lo contrario, false.

Observaciones

Antes de la conexión protegida, el programa deberá establecer una conexión TCP normal con el host con la ayuda de SocketConnect.

Si se da un error al establecer la conexión protegida, en _LastError se registrará el error 5274 (ERR_NETSOCKET_HANDSHAKE_FAILED).

La llamada de esta función no será necesaria si la conexión se realiza con el puerto 443. Se trata del puerto TCP estándar utilizado para las conexiones TLS (SSL) protegidas.

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                            SocketTlsHandshake.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   SERVER    "smtp.gmail.com"
#define   PORT      587
//+------------------------------------------------------------------+
//| Cambio manual de una conexión insegura a una conexión segura     |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool TlsHandshake(int socket)
  {
//--- recibimos el saludo del servidor
   string rsp;

   if(!RecvString(socket,rsp))
      return(false);
//--- saludamos al servidor
   if(!SendString(socket,"EHLO my.domain.com\r\n"))
      return(false);
//--- recibimos una respuesta del servidor con una lista de comandos admitidos
   if(!RecvString(socket,rsp))
      return(false);
//--- imprimimos el saludo
   Print("SERVER: ",rsp);
//--- informamos al servidor de que queremos pasar de una conexión insegura a una segu
   if(!SendString(socket,"STARTTLS\r\n"))
      return(false);
//--- obtenemos la respuesta del servidor
   if(!RecvString(socket,rsp))
      return(false);
//--- En el ejemplo, no comprobamos la respuesta del servidor sobre su preparación par

//--- iniciar una conexión segura TLS (SSL) con el host especificado utilizando el pro
   if(SocketTlsHandshake(socket,SERVER))
      return(true);

   Print("SocketTlsHandshake() failed. Error ",GetLastError());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- creamos un socket y obtenemos su handle
   int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- nos conectamos al servidor SERVER en el puerto PORT
   if(!SocketConnect(socket,SERVER,PORT,10000))
     {
      Print("SocketConnect() failed. Error ",GetLastError());
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- se ha establecido una conexión insegura
      PrintFormat("%s connection has been established to %s:%d",(PORT==443 ? "A secure
      //--- cambiamos a una conexión segura
      if(PORT!=443 && TlsHandshake(socket))
        {
         PrintFormat("Unsecured connection to %s:%d switched to secured",SERVER,PORT);
         //--- si la conexión está protegida por un certificado, mostraremos sus datos
         string   subject,issuer,serial,thumbprint;
         datetime expiration;

         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:      ",subject);
            Print("   Issuer:     ",issuer);
            Print("   Number:     ",serial);
            Print("   Print:      ",thumbprint);
            Print("   Expiration: ",expiration);
           }
        }
     }
//--- cerramos el socket después de usarlo
   SocketClose(socket);
   /*
   resultado:
   An unsecured connection has been established to smtp.gmail.com:587
   SERVER: 220 smtp.gmail.com ESMTP a640c23a62f3a-a9b1f298319sm82305866b.105 - gsmtp

   SERVER: 250-smtp.gmail.com at your service, [37.193.40.122]
   250-SIZE 35882577
   250-8BITMIME
   250-STARTTLS
   250-ENHANCEDSTATUSCODES
   250-PIPELINING
   250-CHUNKING
   250 SMTPUTF8

   SERVER: 220 2.0.0 Ready to start TLS

   SocketTlsHandshake(): A secure connection to smtp.gmail.com:587 is now established
   TLS certificate:
      Owner:  /CN=smtp.gmail.com
      Issuer:  /C=US/O=Google Trust Services/CN=WR2
      Number:     1f:f4:db:2a:5a:e6:dc:52:0a:4c:05:ce:81:cc:c3:f7
      Print: d6be8af229b5329cd3d4c2789c02aa94f89b421c
      Expiration: 2024.12.30 08:25:30
   */
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Envía una cadena al servidor                                     |
//+------------------------------------------------------------------+
bool SendString(int socket,const string str)
  {
//--- convertimos la cadena en un array de caracteres
   uchar data[];
   int   size=StringToCharArray(str,data,0,str.Length(),CP_UTF8);
//--- enviamos los datos al socket
   ResetLastError();
   if(SocketSend(socket,data,size)==size)
      return(true);
//--- error de envío de datos
   Print("Failed to send data to server. Error ",GetLastError());
   return false;
  }
//+------------------------------------------------------------------+
//| Получает строку с сервера                                        |
//+------------------------------------------------------------------+
bool RecvString(int socket,string& result,uint timeout_ms=1000)
  {
//--- esperamos a que los datos aparezcan en el socket
   ulong wait_time_end=GetMicrosecondCount()+timeout_ms*1000;

   while(!SocketIsReadable(socket))
     {
      Sleep(10);
      //--- el tiempo de espera de los datos ha expirado: retornamos NULL como respues
      if(wait_time_end<GetMicrosecondCount())
        {
         Print("ERROR: No response from server");
         return(false);
        }
     }
//--- leemos los datos del socket
   uchar  data[128];
   uint   size=0;
   string resp=NULL;

   do
     {
      uchar b[1];
      int   n=SocketRead(socket,b,1,1000);

      if(n < 0)
         break;

      if(n)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          data[size++]=b[0];

          if(size==data.Size())
            {
             resp += CharArrayToString(data,0,data.Size(),CP_UTF8);
             size = 0;
            }
         }
      }
   while(SocketIsReadable(socket));
//--- copiamos los datos leídos en la cadena
   if(size)
      resp+=CharArrayToString(data,0,size,CP_UTF8);
//--- si la cadena está vacía, tendremos un error
   if(!resp.Length())
     {
      Print("ERROR: No response from server");
      return(false);
     }
//--- entregamos la cadena
   result=resp;
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

SocketTlsCertificate

Obtiene los datos sobre el certificado usado para proteger la conexión de red.

```
bool  SocketTlsCertificate(
   int           socket,              // socket
   string&       subject,             // propietario del certificado
   string&       issuer,              // emisor del certificado
   string&       serial,              // número de serie del certificado
   string&       thumbprint,          // huella digital del certificado
   datetime&     expiration           // plazo de expiración del certificado
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

subject

[in] Nombre del propietario del certificado. Se corresponde con el campo Subject.

issuer

[in] Nombre del emisor del certificado. Se corresponde con el campo Issuer.

serial

[in] Número de serie del certificado. Se corresponde con el campo SerialNumber.

thumbprint

[in] Huella digital del certificado. Se corresponde con el hash SHA-1 del archivo del certificado completo (todos los campos, incluyendo la firma del emisor).

expiration

[in] Plazo de expiración del certificado, en el formato datetime.

Valor retornado

Retorna true en caso de éxito, de lo contrario, false.

Observación

La solicitud de datos del certificado es posible solo después de establecer una conexión protegida con la ayuda de SocketTlsHandshake.

Si se da un error de obtención del certificado, en _LastError se registrará el error 5275 (ERR_NETSOCKET_NO_CERTIFICATE).

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                  SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para que el ejemplo funcione, añada Address a la lista de permi
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos al servidor                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si se usa una conexión TLS protegida a través del puerto 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si se usa una conexión TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- leemos los datos del socket mientras haya, pero no por un tiempo superior a time
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de lectura, dependiendo de si la conexión está o no
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analizamos la respuesta
```

© 2000-2025, MetaQuotes Ltd.

```
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Obtenido el encabezado HTTP de la respuesta:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- comprobando el manejador
   if(socket!=INVALID_HANDLE)
     {
      //--- si todo está en orden, nos conectamos
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conexión establecida con ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la conexión está protegida con un certificado, mostramos sus datos
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Propietario:  ",subject);
            Print("   Emisor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Huella digital: ",thumbprint);
            Print("   Expiración: ",expiration);
            ExtTLS=true;
           }
         //--- enviamos al servidor una solicitud GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitud GET enviada");
            //--- leemos la respuesta
```

© 2000-2025, MetaQuotes Ltd.

```
            if(!HTTPRecv(socket,1000))
               Print("No se ha podido obtener la respuesta, error ",GetLastError());
            }
         else
            Print("No se ha logrado enviar la solicitud GET, error ",GetLastError());
        }
      else
        {
         Print("No se ha logrado conectar con ",Address,":",Port," error ",GetLastErro
        }
      //--- cerramos el socket después de usarlo
      SocketClose(socket);
     }
   else
      Print("No se ha logrado crear el socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketTlsRead

Lee los datos de una conexión TLS protegida.

```
int  SocketTlsRead(
   int           socket,              // socket
   uchar&        buffer[],            // búfer para leer los datos del socket
   uint          buffer_maxlen        // número de bytes que deben ser leídos
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Enlace a la matriz de tipo uchar en la que se leerán los datos. El tamaño de la matriz dinámica aumenta en el número de bytes leídos. El tamaño de la matriz no puede superar INT_MAX (2147483647).

buffer_maxlen

[in] Número de bytes que deben ser leídos en la matriz *buffer[]*. Los datos que no han cabido en la matriz se quedarán en el socket. Será posible obtenerlos con la siguiente llamada de SocketTLSRead. El valor *buffer_maxlen* no puede superar INT_MAX (2147483647).

Valor retornado

En caso de éxito, retorna el número de bytes leídos, en caso de error, retorna -1.

Observación

Si al ejecutar esta función aparece un error en el socket de sistema, la conexión establecida a través de SocketConnect será interrumpida.

La función se ejecuta hasta que obtenga el número indicado de datos o se dé el límite de tiempo (SocketTimeouts).

Si se da un error de lectura de datos, en _LastError se registrará el error 5273 (ERR_NETSOCKET_IO_ERROR).

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version     "1.00"
#property description "Para que el ejemplo funcione, añada Address a la lista de permi
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos al servidor                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si se usa una conexión TLS protegida a través del puerto 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si se usa una conexión TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- leemos los datos del socket mientras haya, pero no por un tiempo superior a time
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de lectura, dependiendo de si la conexión está o no
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analizamos la respuesta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
```

© 2000-2025, MetaQuotes Ltd.

```
               {
                Print("Obtenido el encabezado HTTP de la respuesta:");
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- comprobando el manejador
   if(socket!=INVALID_HANDLE)
     {
      //--- si todo está en orden, nos conectamos
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conexión establecida con ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la conexión está protegida con un certificado, mostramos sus datos
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Propietario:  ",subject);
            Print("   Emisor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Huella digital: ",thumbprint);
            Print("   Expiración: ",expiration);
            ExtTLS=true;
           }
         //--- enviamos al servidor una solicitud GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitud GET enviada");
            //--- leemos la respuesta
            if(!HTTPRecv(socket,1000))
               Print("No se ha podido obtener la respuesta, error ",GetLastError());
           }
         else
            Print("No se ha logrado enviar la solicitud GET, error ",GetLastError());
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      else
        {
         Print("No se ha logrado conectar con ",Address,":",Port," error ",GetLastErro
        }
      //--- cerramos el socket después de usarlo
      SocketClose(socket);
     }
   else
      Print("No se ha logrado crear el socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Mire también

SocketTimeouts, MathSwap

© 2000-2025, MetaQuotes Ltd.

SocketTlsReadAvailable

Lee todos los datos disponibles de una conexión TLS protegida.

```
int  SocketTlsReadAvailable(
   int           socket,              // socket
   uchar&        buffer[],            // búfer para leer los datos del socket
   const uint    buffer_maxlen        // número de bytes que deben ser leídos
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Enlace a la matriz de tipo uchar en la que se leerán los datos. El tamaño de la matriz dinámica aumenta en el número de bytes leídos. El tamaño de la matriz no puede superar INT_MAX (2147483647).

buffer_maxlen

[in] Número de bytes que deben ser leídos en la matriz buffer[]. Los datos que no han cabido en la matriz se quedarán en el socket. Será posible obtenerlos con la siguiente llamada de SocketTlsReadAvailable o SocketTlsRead. El valor de buffer_maxlen no puede superar INT_MAX (2147483647).

Valor retornado

En caso de éxito, retorna el número de bytes leídos, en caso de error, retorna -1.

Observación

Si al ejecutar esta función aparece un error en el socket de sistema, la conexión establecida a través de SocketConnect será interrumpida.

Si se da un error de lectura de datos, en _LastError se registrará el error 5273 (ERR_NETSOCKET_IO_ERROR).

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                      SocketTlsReadAvailable.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property script_show_inputs
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Parámetros de entrada del script                                 |
//+------------------------------------------------------------------+
input string InpMethod ="GET";             // Method (HEAD,GET)
input string InpServer ="www.google.com"; // Server
input uint   InpPort   =443;               // Port
input uint   InpTimeout=5000;              // Timeouts
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
   Print("Server: ",InpServer);
   Print("Port: ",InpPort);
//--- creamos un socket y obtenemos su handle
   const int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- establecemos los timeout para recibir y enviar datos para el objeto de sistema c
   if(!SocketTimeouts(socket,InpTimeout,InpTimeout))
     {
      PrintFormat("SocketTimeouts(%u, %u) failed. Error %d",InpTimeout,InpTimeout,GetL
      SocketClose(socket);
      return;
     }
//--- nos conectamos al servidor Server por el puerto Port
   if(!SocketConnect(socket,InpServer,InpPort,InpTimeout))
     {
      PrintFormat("SocketConnect('%s', %u, %u) failed. Error %d",InpServer,InpPort,Inp
      SocketClose(socket);
      return;
     }
//--- obtenemos los datos sobre el certificado utilizado para proteger la conexión de
   string   subject,issuer,serial,thumbprint;
   datetime expiration;

   if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
     {
      Print("TLS certificate:");
      Print("   Owner:      ",subject);
      Print("   Issuer:     ",issuer);
      Print("   Number:     ",serial);
      Print("   Print:      ",thumbprint);
      Print("   Expiration: ",expiration);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- el servidor no ofrece un certificado: informamos sobre la conexión inseguı
      Print("The connection is not secured by a certificate");
      SocketClose(socket);
      return;
     }
//--- enviamos una solicitud al servidor
   string request=StringFormat("%s / HTTP/1.1\r\nHost: %s\r\nUser-Agent: MetaTrader 5\

   if(HTTPSendTLS(socket,request))
     {
      //--- solicitud enviada: obtenemos una respuesta
      Print("\nRequest sent. Starting page loading...");
      uchar response[]; // todos los datos recibidos (encabezado y cuerpo del document

      if(!HTTPRecvTLS(socket,response,InpTimeout))
        {
         Print("There were errors while reading the page");
         SocketClose(socket);
         return;
        }
      //--- informamos sobre el número de bytes de los datos recibidos
      PrintFormat("%u bytes received",response.Size());
      //--- imprimimos solo el encabezado de la página resultante
      string result    =CharArrayToString(response,0,WHOLE_ARRAY,CP_UTF8);
      int    header_end=StringFind(result,"\r\n\r\n");

      if(header_end>0)
        {
         Print("\nHTTP answer header received:");
         Print(StringSubstr(result,0,header_end));
        }
     }
//--- cerramos el socket después de usarlo
   SocketClose(socket);
   /*
   resultado:
   Server: www.google.com
   Port: 443
   TLS certificate:
      Owner: /CN=www.google.com
      Issuer: /C=US/O=Google Trust Services/CN=WR2
      Number: 0d:43:b1:4a:bb:9c:15:96:10:e1:3d:55:23:9f:25:4e
      Print: 89167618e5017f813aff981c88ce422dc1016bdf
      Expiration: 2024.12.30 08:26:35

   Request sent. Starting page loading...
   HTTPRecvTLS: Document received within 27 attempts
```

© 2000-2025, MetaQuotes Ltd.

```
   25185 bytes received

   HTTP answer header received:
   HTTP/1.1 200 OK
   Date: Fri, 25 Oct 2024 17:12:42 GMT
   Expires: -1
   Cache-Control: private, max-age=0
   Content-Type: text/html; charset=ISO-8859-1
   Content-Security-Policy-Report-Only: object-src 'none';base-uri 'self';script-src '
   Accept-CH: Sec-CH-Prefers-Color-Scheme
   P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
   Server: gws
   X-XSS-Protection: 0
   X-Frame-Options: SAMEORIGIN
   Set-Cookie: AEC=AVYB7coyYMCdweTDTaWeGYzmRnxzKGqsOEosH_VkbCn8xhWkFz6v0kxQFw; expires
   Set-Cookie: NID=518=J02X02Ff4v_9sMcNoUz-1SolmuG08E26Gs438ik0J_SOJUMy7of-P-qup-LaNSW
   Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
   Accept-Ranges: none
   Vary: Accept-Encoding
   Transfer-Encoding: chunked
   */
  }
//+------------------------------------------------------------------+
//| Envía una solicitud HTTP a través de una conexión  segura        |
//+------------------------------------------------------------------+
bool HTTPSendTLS(int socket,const string request)
  {
//--- convertimos la cadena en un array de caracteres, el cero final se descartará
   char req[];
   int  len=StringToCharArray(request,req,0,WHOLE_ARRAY,CP_UTF8)-1;

   if(len<0)
      return false;

   return(SocketTlsSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Recibe una página web a través de una conexión segura            |
//+------------------------------------------------------------------+
bool HTTPRecvTLS(int socket,uchar &response[],const uint timeout_ms)
  {
//--- leemos los datos disponibles de una conexión protegida por TLS antes de que expi
   ulong timeout_check=GetTickCount64()+timeout_ms;
   uchar block[1024];   // búfer de lectura de bloque de los datos del socket
   uint  attempt=0;     // número solicitado de bloques de datos
   int   err    =0;     // код ошибки

   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   do
     {
      //--- leemos por bloques, un máximo de 1024 bytes
      int len=SocketTlsReadAvailable(socket,block,1024);

      if(len>0)
        {
         attempt++;
         //--- unimos los bloques de datos resultantes
         ArrayCopy(response,block,response.Size());
         //--- aquí analizamos los datos recibidos, determinamos el encabezado, el cue
         //...
         //...
         //...
         timeout_check=GetTickCount64()+timeout_ms;
        }
      else
         Sleep(10);

      err=GetLastError();
     }
   while(!IsStopped() && GetTickCount()<timeout_check && !err);
//--- ¿hubo errores en la lectura?
   if(err)
     {
      Print("Error ",err);
      return(false);
     }

   PrintFormat("%s: Document received within %d attempts",__FUNCTION__,attempt);
   return(true);
  }
```

Mire también

SocketTimeouts, MathSwap

© 2000-2025, MetaQuotes Ltd.

SocketTlsSend

Envía los datos a través de una conexión TLS protegida.

```
int  SocketTlsSend(
   int           socket,              // socket
   const uchar&  buffer[],            // búfer para los datos
   uint          buffer_len           // tamaño del búfer
   );
```

Parámetros

socket

[in] Manejador del socket retornado por la función SocketCreate. Al transmitir un manejador incorrecto, en _LastError se registra el error 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[in] Enlace a la matriz de tipo uchar con los datos que deben ser enviados.

buffer_len

[in] Tamaño de la matriz buffer.

Valor retornado

En caso de éxito, retorna el número de bytes registrados en el socket. En caso de error, retorna -1.

Observación

Si al ejecutar esta función aparece un error en el socket de sistema, la conexión establecida a través de SocketConnect será interrumpida.

Si se da un error de registro de datos, en _LastError se registrará el error 5273 (ERR_NETSOCKET_IO_ERROR).

Solo se puede llamar la función desde los expertos y scripts, puesto que funcionan en su propio flujo de ejecución. Si se llama desde el indicador, GetLastError() retornará el error 4014 - "La función de sistema no está permitida para la llamada".

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketTlsSend.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal setting
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =443;
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
   int socket=SocketCreate();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- si todo está en orden, nos conectamos
      if(SocketConnect(socket,Address,Port,1000))
        {
         PrintFormat("Established connection to %s:%d",Address,Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la conexión está protegida por un certificado, mostraremos sus datos
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:      ",subject);
            Print("   Issuer:     ",issuer);
            Print("   Number:     ",serial);
            Print("   Print:      ",thumbprint);
            Print("   Expiration: ",expiration);
           }
         //--- enviamos al servidor una solicitud GET
         string request="GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r\n\r
         char   req[];
         int    len=StringToCharArray(request,req)-1;

         if(len<0)
           {
            Print("StringToCharArray() failed. Error ", GetLastError());
            SocketClose(socket);
            return;
           }
         //--- si se utiliza una conexión protegida por TLS a través del puerto 443
         if(SocketTlsSend(socket,req,len)==len)
           {
            Print("GET request sent");
            //--- leemos la respuesta
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          PrintFormat("Connection to %s:%d failed, error %d",Address,Port,GetLastError
         }
       //--- cerramos el socket después del uso
       SocketClose(socket);
      }
    else
       Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
//| Leyendo la respuesta del servidor                                |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
  {
//--- leemos los datos del socket mientras esté disponible, pero sin superar el timeou
   char   rsp[];
   string result;
   ulong  timeout_check=GetTickCount64()+timeout_ms;

   do
     {
      uint len=SocketIsReadable(socket);

      if(len)
        {
         //--- leemos y analizamos los datos de la conexión TLS segura
         int rsp_len=SocketTlsRead(socket,rsp,len);

         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprimimos solo el encabezado de la respuesta
            int header_end=StringFind(result,"\r\n\r\n");

            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
            //--- actualizamos el tiempo de timeout de la lectura
            timeout_check=GetTickCount64()+timeout_ms;
           }
        }
     }
   while(GetTickCount64()<timeout_check && !IsStopped());

   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

Mire también

SocketTimeouts, MathSwap, StringToCharArray

© 2000-2025, MetaQuotes Ltd.

WebRequest

Envía la solicitud HTTP al servidor especificado. Hay 2 variantes de esta función:

1. **Para el envío de las solicitudes simples** del tipo "clave=valor" usando el encabezado Content-Type: application/x-www-form-urlencoded.

```
int  WebRequest(
   const string      method,           // método HTTP
   const string      url,              // dirección url
   const string      cookie,           // cookie
   const string      referer,          // referer
   int               timeout,          // tiempo de inactividad
   const char        &data[],          // array del cuerpo del mensaje HTTP
   int               data_size,        // tamaño del array data[] bytes
   char              &result[],        // array con los datos de respuesta del servido
   string            &result_headers   // encabezado de la respuesta del servidor
   );
```

2. **Para el envío de las solicitudes del tipo libre** especificando el conjunto individual de encabezados para la interacción más flexible con diferentes servicios Web.

```
int  WebRequest(
   const string      method,           // método HTTP
   const string      url,              // dirección url
   const string      headers,          // encabezados
   int               timeout,          // tiempo de inactividad
   const char        &data[],          // array del cuerpo del mensaje HTTP
   char              &result[],        // array con los datos de respuesta del servido
   string            &result_headers   // encabezado de la respuesta del servidor
   );
```

Parámetros

method

[in] Método HTTP.

url

[in] Dirección URL.

headers

[in] Encabezados de la solicitud del tipo "clave: valor" separados con salto de línea "\r\n".

cookie

[in] Valor Cookie.

referer

[in] Valor del encabezado de la solicitud Referer HTTP.

timeout

[in] Tiempo de inactividad en milisegundos.

© 2000-2025, MetaQuotes Ltd.

data[]

[in] Array de datos del cuerpo del mensaje HTTP.

data_size

[in] Tamaño del array data[].

result[]

[out] Array con los datos de respuesta del servidor.

result_headers

[out] Encabezados de la respuesta del servidor.

Valor devuelto

Código de la respuesta del servidor HTTP, o -1 en caso del error.

Nota

Para el uso de la función WebRequest() hay que añadir las direcciones de los servidores en la lista de las URLs permitidas en la pestaña "Asesores Expertos" de la ventana "Ajustes". El puerto del servidor se selecciona automáticamente a base del protocolo especificado: 80 para "http://" y 443 para "https://".

La función WebRequest() es sincrónica. Eso significa que interrumpe la ejecución del programa y espera la respuesta del servidor solicitado. Puesto que los retardos a la hora de recibir la respuesta a la solicitud enviada pueden ser importantes, esta función está prohibida para las llamadas desde los indicadores porque los indicadores trabajan en el único flujo común para todos los indicadores y gráficos de este símbolo. El retardo de ejecución del indicador en uno de los gráficos del símbolo puede provocar la parada de las actualizaciones de todos los gráficos de este símbolo.

Esta función puede invocarse sólo desde los EAs y los scripts porque trabajan en su propio flujo de ejecución. Si la función se invoca desde un indicador, GetLastError() devolverá el error 4014 - "Función prohibida".

Durante el trabajo en el Probador de Estrategias la función WebRequest() no se ejecuta.

Ejemplo:

```
void OnStart()
  {
   string cookie=NULL,headers;
   char   post[],result[];
   string url="https://finance.yahoo.com";
//--- para trabajar con el servidor es necesario añadir el URL "https://finance.yahoo.
//--- en la lista de URL permitidos (Menú principal->Servicio->Ajustes, pestaña "Asesc
//--- reseteamos el código del último error
   ResetLastError();
//--- carga de la página html desde Yahoo Finance
   int res=WebRequest("GET",url,cookie,NULL,500,post,0,result,headers);
   if(res==-1)
     {
      Print("Error en WebRequest. Código de error  =",GetLastError());
      //--- es posible que el URL no esté en la lista; mostramos un mensaje que indiqu
```

© 2000-2025, MetaQuotes Ltd.

```
      MessageBox("Es necesario añadir la dirección '"+url+"' a la lista de URL permiti
     }
   else
     {
      if(res==200)
        {
         //--- la carga se ha realizado con éxito
         PrintFormat("El archivo se ha cargado con éxito, tamaño %d bytes.",ArraySize
         //PrintFormat("Encabezados del servidor: %s",headers);
         //--- guardamos los datos en el archivo
         int filehandle=FileOpen("url.htm",FILE_WRITE|FILE_BIN);
         if(filehandle!=INVALID_HANDLE)
           {
            //--- guardamos el contenido del array result[] en el archivo
            FileWriteArray(filehandle,result,0,ArraySize(result));
            //--- cerramos el archivo
            FileClose(filehandle);
           }
         else
            Print("Error en FileOpen. Código de error =",GetLastError());
        }
      else
         PrintFormat("Error de carga '%s', código %d",url,res);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SendFTP

Envía un archivo a la dirección especificada en la ventana de configuraciones, en la pestaña "Edición".

```
bool  SendFTP(
   string  filename,          // archivo a mandar a través de ftp
   string  ftp_path=NULL      // ruta para la descarga en el servidor-ftp
   );
```

Parámetros

filename

[in] Nombre del archivo a mandar.

ftp_path=NULL

[in] Directorio FTP. Si no está especificado, se utiliza el directorio descrito en las configuraciones.

Valor devuelto

En caso de fallo devuelve false.

Nota

El archivo a mandar tiene que estar ubicado en la carpeta *directorio_de_terminal\MQL5\files* o en sus subcarpetas. El archivo no se envía si la dirección FTP y/o la contraseña de acceso no están especificados en las configuraciones.

Durante el trabajo en el Probador de Estrategias la función SendFTP() no se ejecuta.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                     SendFTP.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   FILENAME  "SomeFile.bin"
#define   PATH      NULL
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- comprobamos el permiso en el terminal para enviar archivos a la dirección FTP
   if(!TerminalInfoInteger(TERMINAL_FTP_ENABLED))
     {
      Print("Error. The client terminal does not have permission to send messages to t
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- enviamos el archivo
   ResetLastError();
   if(!SendFTP(FILENAME, PATH))
      Print("SendFTP() failed. Error ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

SendMail

Envía una carta electrónica a la dirección especificada en la ventana de configuraciones, en la pestaña "Edición".

```
bool  SendMail(
   string  subject,        // asunto
   string  some_text       // texto de la carta
   );
```

Parámetros

subject

[in] Asunto de la carta.

some_text

[in] Cuerpo de la carta.

Valor devuelto

true - si la carta ha sido puesta en cola del envío, en caso contrario devuelve false.

Nota

El envío puede estar prohibido por la configuración, también puede faltar la dirección del correo electrónico. Para obtener la información sobre el error hay que llamar a la función GetLastError().

Durante el trabajo en el Probador de Estrategias la función SendMail() no se ejecuta.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                     SendMail.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   SUBJECT   "Test SendMail"
#define   TEXT      "Text for SendMail() function"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- comprobamos el permiso para enviar un correo electrónico en el terminal
   if(!TerminalInfoInteger(TERMINAL_EMAIL_ENABLED))
     {
      Print("Error. The client terminal does not have permission to send email message
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- enviamos la carta
   ResetLastError();
   if(!SendMail(SUBJECT, TEXT))
      Print("SendMail() failed. Error ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

SendNotification

Esta función envía avisos a los terminales móviles cuyos MetaQuotes ID han sido indicados en la ventana de ajustes en la pestaña "Notificaciones".

```
bool  SendNotification(
   string  text          // texto del mensaje
   );
```

Parámetros

text

[in] Texto del mensaje en la notificación. La longitud del mensaje no puede superar 255 caracteres.

Valor devuelto

Devuelve true si la notificación ha sido enviada con éxito desde el terminal; de lo contrario, devuelve false. Durante la comprobación tras el fallo del envío, GetLastError() puede mostrar uno de los siguientes errores:

- 4515 - ERR_NOTIFICATION_SEND_FAILED,
- 4516 - ERR_NOTIFICATION_WRONG_PARAMETER,
- 4517 - ERR_NOTIFICATION_WRONG_SETTINGS,
- 4518 - ERR_NOTIFICATION_TOO_FREQUENT.

Nota

Para la función SendNotification() existen unas estrictas limitaciones de uso: no más de dos llamadas al segundo y no más de 10 llamadas al minuto. La frecuencia de uso se controla de forma dinámica, y la función puede ser bloqueada si tiene lugar la infracción de estas condiciones.

Durante el trabajo en el Probador de Estrategias la función SendNotification() no se ejecuta.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                             SendNotification.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   MESSAGE   "Test Message"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- comprobamos el permiso de envío de notificaciones en el terminal
   if(!TerminalInfoInteger(TERMINAL_NOTIFICATIONS_ENABLED))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("Error. The client terminal does not have permission to send notifications
       return;
      }
//--- enviamos una notificación
   ResetLastError();
   if(!SendNotification(MESSAGE))
      Print("SendNotification() failed. Error ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

Variables globales del terminal de cliente

Conjunto de funciones para trabajar con variables globales.

No se puede confundir las variables globales del terminal de cliente con las variables declaradas a nivel global del programa mql5.

Las variables globales se guardan en el terminal de cliente durante 4 semanas desde el último acceso, luego se eliminan automáticamente. El acceso a una variable global no es sólo la difinición de un valor nuevo, sino también la lectura del valor de una variable global.

Las variables globales del terminal de cliente están disponibles al mismo tiempo desde todos los programas mql5 abiertas en el terminal de cliente.

Función	Acción
GlobalVariableCheck	Comprueba la existencia de una variable global con el nombre especificado
GlobalVariableTime	Devuelve la hora del último acceso a una variable global
GlobalVariableDel	Elimina una variable global
GlobalVariableGet	Solicita el valor de una variable global
GlobalVariableName	Devuelve el nombre de una variable global según su número ordinal en la lista de variables globales
GlobalVariableSet	Establece el nuevo valor para una variable global
GlobalVariablesFlush	Guarda por vía forzada el contenido de todas las variables globales en el disco
GlobalVariableTemp	Establece el nuevo valor para una variable global que existe sólo durante esta sesión del terminal
GlobalVariableSetOnCondition	Establece el nuevo valor de una variable global ya existente según una condición
GlobalVariablesDeleteAll	Elimina variables globales con el prefijo especificado en su nombre
GlobalVariablesTotal	Devuelve la cantidad total de variables globales

© 2000-2025, MetaQuotes Ltd.

GlobalVariableCheck

Comprueba la existencia de una variable global del terminal de cliente.

```
bool  GlobalVariableCheck(
   string  name      // nombre
   );
```

Parámetros

name

[in] Nombre de la variable global.

Valor devuelto

Devuelve el valor true, si la variable global existe, de lo contrario devuelve false.

Nota

Las variables globales se guardan en el terminal de cliente durante 4 semanas desde el último acceso, luego se eliminan automáticamente.

Véase también

GlobalVariableTime()

© 2000-2025, MetaQuotes Ltd.

GlobalVariableTime

Devuelve la hora del último acceso a una variable global.

```
datetime  GlobalVariableTime(
   string  name      // nombre
   );
```

Parámetros

name

[in] Nombre de la variable global.

Valor devuelto

Devuelve la hora del último acceso a la variable global especificada. El direccionamiento a la variable por su valor, por ejemplo, usando las funciones GlobalVariableGet() y GlobalVariableCheck(), también cambia la hora del último acceso. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Las variables globales se guardan en el terminal de cliente durante 4 semanas desde el último acceso, luego se eliminan automáticamente.

Véase también

GlobalVariableCheck()

© 2000-2025, MetaQuotes Ltd.

GlobalVariableDel

Elimina una variable global del terminal de cliente.

```
bool  GlobalVariableDel(
   string  name      // nombre
   );
```

Parámetros

name

[in] Nombre de la variable global.

Valor devuelto

Si se elimina con éxito, la función devuelve true, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Las variables globales se guardan en el terminal de cliente durante 4 semanas desde el último acceso, luego se eliminan automáticamente.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableGet

Devuelve el valor de una variable global ya existente del terminal de cliente. Existen 2 variantes de la función.

1. Devuelve el valor de la propiedad directamente.

```
double  GlobalVariableGet(
   string  name      // nombre
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso del éxito el valor de una variable global se coloca en una variable receptora que es pasada por referencia por el segundo parámetro.

```
bool  GlobalVariableGet(
   string  name               // nombre
   double& double_var         // aquí recibimos el valor de la variable global
   );
```

Parámetros

name

[in] Nombre de la variable global.

double_var

[out] Variable del tipo double que recibe el valor guardado en una variable global del terminal de cliente.

Valor devuelto

El valor de la variable global existente o 0 en caso del error. Para obtener la información sobre el error hay que llamar a la función GetLastError().

Nota

Las variables globales se guardan en el terminal de cliente durante 4 semanas desde el último acceso, luego se eliminan automáticamente.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableName

Devuelve el nombre de una variable global según su número ordinal.

```
string  GlobalVariableName(
   int  index      // número en la lista de variables globales
   );
```

Parámetros

index

[in] Número ordinal en la lista de variables globales. Tiene que ser más de o igual a 0 y menos de GlobalVariablesTotal() .

Valor devuelto

Nombre de una variable global según su número ordinal en la lista de variables globales. Para obtener la información sobre el error hay que llamar a la función GetLastError().

Nota

Las variables globales se guardan en el terminal de cliente durante 4 semanas desde el último acceso, luego se eliminan automáticamente.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableSet

Establece el nuevo valor para una variable global. Si la variable no existe, el sistema creará una nueva variable global.

```
datetime  GlobalVariableSet(
   string  name,      // nombre
   double  value      // valor que se asigna
   );
```

Parámetros

name

[in] Nombre de la variable global.

value

[in] Valor numérico nuevo.

Valor devuelto

Si se ejecuta con éxito la función devuelve la hora del último acceso, de lo contrario devuelve 0. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

El nombre de la variable global no debe superar 63 caracteres. Las variables globales se guardan en el terminal de cliente durante 4 semanas desde el último acceso, luego se eliminan automáticamente.

© 2000-2025, MetaQuotes Ltd.

GlobalVariablesFlush

Guarda por vía forzada el contenido de todas las variables globales en el disco.

```
void  GlobalVariablesFlush();
```

Valor devuelto

No hay valor devuelto.

Nota

El terminal guarda todas las variables globales durante la finalización de su trabajo pero con un fallo repentino del ordenador los datos pueden perderse. Esta función permite dirigir el proceso de guardado de variables globales de una manera independiente en caso de una emergencia.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableTemp

Intenta crear una nueva variable global. Si la variable no existe, el sistema creará una nueva variable global temporal.

```
bool  GlobalVariableTemp(
   string  name,      // nombre
   );
```

Parámetros

name

[in] Nombre de la variable global temporal.

Valor devuelto

En caso de éxito la función devuelve true, de lo contrario devuelve false. Para obtener la información sobre el error hay que llamar a la función GetLastError().

Nota

Las variables globales temporales existen sólo durante el funcionamiento del terminal de cliente, después del cierre del terminal estas variables se eliminan automáticamente. Durante la ejecución de la operación GlobalVariablesFlush() las variables globales temporales no se guardan en el disco.

Una vez creada una variable global temporal, el acceso a ella y su modificación se realizan igual que a una variable global del terminal de cliente común.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableSetOnCondition

Establece un nuevo valor de una variable global existente, si el valor actual de la variable es igual al valor del tercer parámetro check_value. Si la variable no existe, la función generará el error ERR_GLOBALVARIABLE_NOT_FOUND (4501) y devolverá false.

```
bool  GlobalVariableSetOnCondition(
   string  name,            // nombre
   double  value,           // valor si se cumple la condición
   double  check_value      // condición que se comprueba
   );
```

Parámetros

name

[in] Nombre de la variable global.

value

[in] Valor nuevo.

check_value

[in] Valor para comprobar el valor actual de la variable global.

Valor devuelto

En caso de éxito la función devuelve true, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError(). Si el valor actual de la variable global es diferente a check_value, la función devolverá false.

Nota

La función proporciona el acceso atómico a una variable global, por eso se puede usarla para organizar un mutex en caso de interacción de varios Asesores Expertos que trabajan al mismo tiempo dentro de un terminal de cliente.

© 2000-2025, MetaQuotes Ltd.

GlobalVariablesDeleteAll

Elimina variables globales del terminal de cliente.

```
int  GlobalVariablesDeleteAll(
   string     prefix_name=NULL      // todas las variables globales cuyos nombres se e
   datetime   limit_data=0          // todas las variables globales que han sido cambi
   );
```

Parámetros

prefix_name=NULL

[in] Prefijo del nombre de las variables globales que se eliminan. Si se trata del prefijo NULL o cadena vacía, entonces todas las variables globales que cumplen con el criterio de la fecha serán eliminadas.

limit_data=0

[in] Fecha para la selección de variables globales según el momento de su última modificación. Las variables globales que se han modificado antes de la fecha indicada serán eliminadas. Si el parámetro es igual a cero, se eliminan todas las variables globales que cumplen con el primer criterio (según el prefijo).

Valor devuelto

Número de variables eliminadas.

Nota

Si ambos parámetros son iguales a cero (prefix_name=NULL y limit_data=0), se eliminan todas las variables globales del terminal. Si los dos parámetros están especificados, se eliminan las variables globales correspondientes a cada uno de los parámetros especificados.

Las variables globales se guardan en el terminal de cliente durante 4 semanas desde el último acceso, luego se eliminan automáticamente.

© 2000-2025, MetaQuotes Ltd.

GlobalVariablesTotal

Devuelve la cantidad total de variables globales del terminal de cliente.

```
int  GlobalVariablesTotal();
```

Valor devuelto

Número de variables globales.

Nota

Las variables globales se guardan en el terminal de cliente durante 4 semanas desde el último acceso, luego se eliminan automáticamente. El acceso a una variable global no es sólo la difinición de un valor nuevo, sino también la lectura del valor de una variable global.

© 2000-2025, MetaQuotes Ltd.

Operaciones con archivos

Éste es el grupo de funciones que se utilizan para operar con los archivos.

Existen dos carpetas (con subcarpetas) en las que se puede colocar los archivos de trabajo:

- terminal_data_directorio\MQL5\FILES\ (para verla seleccione el punto del menú "Archivo"-"Abrir carpeta de datos" en el terminal);
- carpeta general de todos los terminales instalados en el ordenador - suele ubicarse en el directorio C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal\Common\Files.

Se puede obtener los nombres de estos catálogos de forma de programación empleando la función TerminalInfoString() y usando las enumeraciones ENUM_TERMINAL_INFO_STRING:

```
//--- Carpeta en la que se guardan los datos del terminal
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- Directorio general de todos los terminales de cliente
   string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

Está prohibido trabajar con los archivos desde otras carpetas.

Las funciones de archivos permiten trabajar con lo que llaman "tuberías nombradas". Para eso sólo hay que llamar a la función FileOpen() con los parámetros correspondientes.

Función	Acción
FileSelectDialog	Crea una ventana de diálogo para crear/abrir un archivo o carpeta
FileFindFirst	Empieza la búsqueda de los archivos en el directorio correspondiente de acuerdo con el filtro especificado
FileFindNext	Sigue con la búsqueda empezada por la función FileFindFirst()
FileFindClose	Cierra el manejador de búsqueda
FileOpen	Abre un archivo con el nombre y banderas especificados
FileDelete	Elimina un archivo especificado
FileFlush	Guarda en el disco todos los datos que se han quedado en el buffer de entrada/salida
FileGetInteger	Obtiene una propiedad del número entero del archivo
FileIsEnding	Determina el final de un archivo en el proceso de lectura
FileIsLineEnding	Determina el fin de una línea en un archivo de texto en el proceso de lectura
FileClose	Cierra un archivo previamente abierto
FileIsExist	Comprueba la existencia de un archivo
FileCopy	Copia el archivo original de una carpeta local o compartida a otro archivo
FileMove	Mueve o renombra un archivo

© 2000-2025, MetaQuotes Ltd.

Función	Acción
FileReadArray	Lee los arrays de cualquier tipo, salvo los arrays literales (string) (puede ser un array de estructuras que no contienen las cadenas ni arrays dinámicos) de un archivo binario desde la posición actual del puntero de archivos
FileReadBool	Lee de un archivo del tipo CSV una cadena desde la posición actual hasta el separador (o hasta el final de la línea de texto) y convierte la cadena leída al valor del tipo bool
FileReadDatetime	Lee de un archivo del tipo CSV una cadena de uno de los formatos: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" o "HH:MI:SS" - y la convierte al valor del tipo datetime
FileReadDouble	Lee un número de doble precisión con punto flotante (double) de un archivo binario desde la posición actual del puntero de archivos
FileReadFloat	Lee desde la posición actual del puntero de archivos valor del tipo float
FileReadInteger	Lee de un archivo binario valor del tipo int, short o char dependiendo de la longitud indicada en bytes
FileReadLong	Lee desde la posición actual del puntero de archivos valor del tipo long
FileReadNumber	Lee de un archivo del tipo CSV una cadena desde la posición actual hasta el separador (o hasta el final de la línea de texto) y convierte la cadena leída al valor del tipo double
FileReadString	Lee de un archivo una cadena desde la posición actual del puntero de archivos
FileReadStruct	Lee de un archivo binario el contenido en una estructura que ha sido pasada como un parámetro
FileSeek	Mueve la posición del puntero de archivos a una cantidad de bytes especificada respecto a la posición indicada
FileSize	Devuelve el tamaño de un archivo correspondiente abierto
FileTell	Devuelve la posición actual del puntero de archivos de un archivo correspondiente abierto
FileWrite	Escribe los datos en un archivo del tipo CSV o TXT
FileWriteArray	Escribe los arrays de cualquier tipo (excepto los arrays string) en un archivo del tipo BIN
FileWriteDouble	Escribe el valor del parámetro del tipo double desde la posición actual del puntero de archivos en un archivo binario
FileWriteFloat	Escribe el valor del parámetro del tipo float desde la posición actual del puntero de archivos en un archivo binario

© 2000-2025, MetaQuotes Ltd.

Función	Acción
FileWriteInteger	Escribe el valor del parámetro del tipo int desde la posición actual del puntero de archivos en un archivo binario
FileWriteLong	Escribe el valor del parámetro del tipo long desde la posición actual del puntero de archivos en un archivo binario
FileWriteString	Escribe el valor del parámetro del tipo string desde la posición actual del puntero de archivos en un archivo del tipo BIN o TXT
FileWriteStruct	Escribe el contenido de una estructura pasada como un parámetro en un archivo binario desde la posición actual del puntero de archivos
FileLoad	Lee todo el contenido de un archivo binario especificado en una matriz transmitida de tipos numéricos o estructuras sencillas
FileSave	Guarda en un archivo binario todos los elementos de la matriz transmitida como parámetro
FolderCreate	Crea un directorio en la carpeta Files (dependiendo del valor common_flag)
FolderDelete	Elimina un directorio seleccionado. Una carpeta no vacía no puede ser eliminada
FolderClean	Elimina todos los archivos en la carpeta especificada

Si el archivo se abre para escritura usando la función FileOpen(), todas las subcarpetas indicadas en la ruta van a ser creadas en caso si no existen.

© 2000-2025, MetaQuotes Ltd.

FileSelectDialog

Crea una ventana de diálogo para crear/abrir un archivo o carpeta.

```
int  FileSelectDialog(
   string   caption,              // encabezado de la ventana
   string   initial_dir,          // carpeta inicial
   string   filter,               // filtro de extensiones
   uint     flags,                // combinación de banderas
   string&  filenames[],          // matriz con los nombres de los archivos
   string   default_filename      // nombre del archivo por defecto
   );
```

Parámetros

caption

[in] Encabezado de la ventana de diálogo.

initial_dir

[in] Nombre de la carpeta inicial con respecto a la carpeta MQL5\Files cuyo contenido se mostrará en la ventana de diálogo. Si el valor es igual a NULL, en la ventana de diálogo se mostrará la carpeta MQL5\Files.

filter

[in] Filtro de extensiones de los archivos que se mostrarán en la ventana de diálogo para su selección. Los archivos de otros formatos permanecerán ocultos.

flags

[in] Combinación de banderas que determina el modo de la ventana de diálogo. Las banderas se determinan de la forma siguiente:
FSD_WRITE_FILE - ventana de diálogo de apertura de archivos;
FSD_SELECT_FOLDER - permite seleccionar solo carpetas;
FSD_ALLOW_MULTISELECT - permite seleccionar varios archivos;
FSD_FILE_MUST_EXIST - los archivos seleccionados deben existir;
FSD_COMMON_FOLDER - el archivo está ubicado en la carpeta general de todos los terminales de cliente \Terminal\Common\Files.

filenames[]

[out] Matriz de líneas en la que se ubicarán los nombres de los archivos/carpetas elegidos.

default_filename

[in] Nombre del archivo/carpeta por defecto. Si ha sido establecido, este nombre se añade automáticamente a la ventana de diálogo de apertura, y se retorna en la matriz *filenames[]* en la simulación.

Valor retornado

Si se finaliza con éxito, la función retornará el número de archivos seleccionados cuyos nombres se pueden obtener en *filenames[]*. Si el usuario no ha seleccionado un archivo y ha cerrado la ventana de diálogo, la función retornará 0. Si se ejecuta sin éxito, se retornará un valor inferior a 0, el código de error se puede obtener con la ayuda de GetLastError().

© 2000-2025, MetaQuotes Ltd.

Observación

Por motivos de seguridad, en el lenguaje MQL5 se controla estrictamente el trabajo con los archivos. Los archivos con los que se realizan operaciones de archivo usando los recursos de MQL5, no pueden encontrarse fuera del "sandbox" de archivo, para ser más exactos, fuera de la carpeta MQL5\Files.

El nombre de la carpeta inicial *initial_dir* se busca en el terminal de cliente, en la subcarpeta MQL5\Files (o catálogo_de_agente_de_simulación\MQL5\Files, en el caso de simulación). Si entre las banderas se indica FSD_COMMON_FOLDER, la carpeta inicial se buscará en la carpeta general de todos los terminales de cliente \Terminal\Common\Files.

El parámetro *filter* indica los archivos permitidos, y debe indicarse en el formato "<descripción 1>|<extensión 1>|<descripción 2>|<extensión 2>...". Por ejemplo, "Text files (*.txt)|*.txt|All files (*.*)|*.*", además, en este caso, la primera extensión "Text files (*.txt)|*.txt" se seleccionará como tipo de archivo por defecto.
Si *filter*=NULL, la máscara de selección de archivos en la ventana de diálogo será "All Files (*.*)|*.*|"

Si se ha establecido el parámetro *default_filename*, durante la simulación no visual, la llamada de FileSelectDialog() retornará 1, mientras que el propio valor *default_filename* se copiará en la matriz *filenames[]*.

La función está prohibida en los indicadores de usuario, pues la llamada de FileSelectDialog() interrumpe el funcionamiento del flujo de ejecución mientras se espera la respuesta del usuario. Y puesto que todos los indicadores de cada símbolo se ejecutan en un único flujo, semejante interrupción hará imposible el funcionamiento de todos los gráficos en todos los marcos temporales de este símbolo.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenemos los nombres de los archivos de texto para realizar la carga desde la c
   string filenames[];
   if(FileSelectDialog("Seleccione los archivos para la carga", NULL,
                       "Text files (*.txt)|*.txt|All files (*.*)|*.*",
                       FSD_ALLOW_MULTISELECT|FSD_COMMON_FOLDER, filenames, "data.txt")
     {
      //--- mostramos el nombre de cada archivo seleccionado
      int total=ArraySize(filenames);
      for(int i=0; i<total; i++)
         Print(i, ": ", filenames[i]);
     }
   else
     {
     Print("Files not selected");
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

Ver también

FileOpen, FileIsExist, FileDelete, FileMove, FolderCreate, FolderDelete, FolderClean, banderas de apertura de archivos

© 2000-2025, MetaQuotes Ltd.

FileFindFirst

La función empieza la búsqueda de los archivos y subcarpetas en el directorio correspondiente de acuerdo con el filtro especificado.

```
long  FileFindFirst(
   const string   file_filter,          // cadena - filtro de búsqueda
   string&        returned_filename,    // nombre del archivo o subcarpeta encontrada
   int            common_flag=0         // determina la zona de búsqueda
   );
```

Parámetros

file_filter

[in] Filtro de búsqueda. En el filtro se puede especificar un subdirectorio (o una sucesión de subdirectorios anidados) respecto al directorio \Files en el que se precisa realizar la búsqueda.

returned_filename

[out] Parámetro devuelto en el que se coloca el nombre del primer archivo o subcarpeta encontrada en caso del éxito. Only the file name is returned (including the extension), the directories and subdirectories are not included no matter if they are specified or not in the search filter.

common_flag

[in] Bandera que determina la ubicación del archivo. Si common_flag=FILE_COMMON, entonces el archivo se encuentra en una carpeta compartida de todos los terminales de cliente \Terminal\Common\Files. De lo contrario, el archivo está ubicado en una carpeta local.

Valor devuelto

Devuelve el manejador del objeto de búsqueda que hay que utilizar para la siguiente búsqueda de archivos y subcarpetas por la función FileFindNext(), o INVALID_HANDLE en caso no hay ningún archivo o subcarpeta que corresponda al filtro (en caso particular - la subcarpeta está vacía). Después de finalizar la búsqueda, hay que cerrar el manejador usando la función FileFindClose().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//--- display the window of input parameters when launching the script
#property script_show_inputs
//--- filter
input string InpFilter="Dir1\\*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   string int_dir="";
   int    i=1,pos=0,last_pos=-1;
//--- search for the last backslash
   while(!IsStopped())
     {
      pos=StringFind(InpFilter,"\\",pos+1);
      if(pos>=0)
         last_pos=pos;
      else
         break;
     }
//--- the filter contains the folder name
   if(last_pos>=0)
      int_dir=StringSubstr(InpFilter,0,last_pos+1);
//--- get the search handle in the root of the local folder
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- check if the FileFindFirst() is executed successfully
   if(search_handle!=INVALID_HANDLE)
     {
      //--- in a cycle, check if the passed strings are the names of files or directo
      do
        {
         ResetLastError();
         //--- if it's a file, the function returns true, and if it's a directory, it
         FileIsExist(int_dir+file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- close the search handle
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

Véase también

FileFindNext, FileFindClose

© 2000-2025, MetaQuotes Ltd.

FileFindNext

Sigue con la búsqueda empezada por la función FileFindFirst().

```
bool  FileFindNext(
   long      search_handle,         // manejador de búsqueda
   string&   returned_filename      // nombre del archivo o subcarpeta encontrada
   );
```

Parámetros

search_handle

[in] Manejador de búsqueda recibido de la función FileFindFirst().

returned_filename

[out] Nombre del siguiente archivo o subcarpeta encontrada. Only the file name is returned (including the extension), the directories and subdirectories are not included no matter if they are specified or not in the search filter.

Valor devuelto

En caso del éxito devuelve true, de lo contrario devuelve false.

Ejemplo:

```
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- filtro
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- obtenemos el manejador de búsqueda en la raíz de la carpeta local
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- comprobamos si la función FileFindFirst() se ha ejecutado con éxito
   if(search_handle!=INVALID_HANDLE)
     {
      //--- comprobamos en el ciclo si las cadenas pasadas son los nombres de los arch
      do
        {
         ResetLastError();
         //--- si es un archivo, la función devuelve true, y si es una carpeta, la fur
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- cerramos el manejador de la búsqueda
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

© 2000-2025, MetaQuotes Ltd.

Véase también

FileFindFirst, FileFindClose

© 2000-2025, MetaQuotes Ltd.

FileFindClose

Cierra el manejador de búsqueda.

```
void  FileFindClose(
   long  search_handle      // manejador de búsqueda
   );
```

Parámetros

search_handle

[in] Manejador de búsqueda recibido de la función FileFindFirst().

Valor devuelto

No hay valor devuelto.

Nota

Hay que llamar a la función para liberar los recursos del sistema.

Ejemplo:

```
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- filtro
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- obtenemos el manejador de búsqueda en la raíz de la carpeta local
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- comprobamos si la función FileFindFirst() se ha ejecutado con éxito
   if(search_handle!=INVALID_HANDLE)
     {
      //--- comprobamos en el ciclo si las cadenas pasadas son los nombres de los arch
      do
        {
         ResetLastError();
         //--- si es un archivo, la función devuelve true, y si es una carpeta, la fun
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==5018 ? "Directory" : "File"
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- cerramos el manejador de la búsqueda
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

Véase también

FileFindFirst, FileFindNext

© 2000-2025, MetaQuotes Ltd.

FileIsExist

Esta función comprueba la existencia de un archivo.

```
bool  FileIsExist(
   const string  file_name,        // nombre del archivo
   int           common_flag=0     // zona de búsqueda
   );
```

Parámetros

file_name

[in] Nombre del archivo a comprobar.

common_flag=0

[in] Bandera que determina la ubicación del archivo. Si common_flag=FILE_COMMON, entonces el archivo se encuentra en una carpeta compartida de todos los terminales de cliente \Terminal\Common\Files. De lo contrario, el archivo está ubicado en una carpeta local.

Valor devuelto

Devuelve true, si el archivo especificado existe.

Nota

El archivo comprobado puede resultar una subcarpeta. En este caso la función FileIsExist() devuelve false y en la variable _LastError se inscribe el error 5018 - "No es un archivo, sino una subcarpeta" (ver el ejemplo para la función FileFindFirst).

Por razones de seguridad el trabajo con los archivos en el lenguaje MQL5 se encuentra bajo un estricto control. Los archivos con los que se realizan las operaciones de archivos utilizando los medios del lenguaje MQL5 no pueden estar fuera del entorno protegido de archivos (file sandbox).

Si common_flag=FILE_COMMON, la función busca el archivo especificado en una carpeta compartida de todos los terminales de cliente \Terminal\Common\Files, de lo contrario la función lo busca en una carpeta local (MQL5\Files o MQL5\Tester\Files en caso de prueba).

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- fecha para archivos antiguos
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // variable para almacenar los nombres de archivos
   string   filter="*.txt"; // filtro para la búsqueda de archivos
   datetime create_date;    // fecha de creación del archivo
   string   files[];        // lista con los nombres de los archivos
   int      def_size=25;    // tamaño del array por defecto
   int      size=0;         // número de archivos
//--- adjudicar memoria para array
   ArrayResize(files,def_size);
//--- recibir el manejador de búsqueda en la raíz de la carpeta local
   long search_handle=FileFindFirst(filter,file_name);
//--- comprobamos si la función FileFindFirst() se ha ejecutado con exito
   if(search_handle!=INVALID_HANDLE)
     {
      //--- repasamos los archivos en el ciclo
      do
        {
         files[size]=file_name;
         //--- aumentamos el tamaño del array
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- reseteamos el valor del error
         ResetLastError();
         //--- obtenemos la fecha de creación del archivo
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- comprobamos si el archivo es antiguo
         if(create_date<InpFilesDate)
           {
            PrintFormat("¡El archivo %s ha sido eliminado!",file_name);
            //--- eliminamos el archivo antiguo
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- cerramos el manejador de búsqueda
      FileFindClose(search_handle);
     }
   else
     {
      Print("Files not found!");
      return;
     }
//--- comprobamos qué archivos se han quedado
   PrintFormat("Resultados:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("¡El archivo %s existe!",files[i]);
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("¡El archivo %s ha sido eliminado!",files[i]);
      }
   }
```

Véase también

FileFindFirst

© 2000-2025, MetaQuotes Ltd.

FileOpen

La función abre el archivo con el nombre especificado y las banderas especificadas.

```
int  FileOpen(
   string  file_name,           // nombre del archivo
   int     open_flags,          // combinación de banderas
   short   delimiter='\t',      // delimitador
   uint    codepage=CP_ACP      // página de código
   );
```

Parámetros

file_name

[in] Nombre del archivo a abrir, puede contener subcarpetas. Si el archivo se abre para la escritura, las subcarpetas especificadas serán creadas en caso de que no existan.

open_flags

[in] combinación de banderas que determina el modo de trabajo con el archivo. Las banderas están definidas como sigue:
FILE_READ el archivo se abre para la lectura
FILE_WRITE el archivo se abre para la escritura
FILE_BIN modo binario de lectura-escritura (sin conversión de una cadena, ni tampoco en una cadena)
FILE_CSV archivo del tipo csv (todos los elementos grabados se convierten a las cadenas del tipo correspondiente, unicode o ansi, y se separan con un delimitador)
FILE_TXT archivo de texto simple (igual que el archivo csv pero sin tomar en cuenta los delimitadores)
FILE_ANSI cadenas del tipo ANSI (símbolos de un byte)
FILE_UNICODE cadenas del tipo UNICODE (símbolos de dos bytes)
FILE_SHARE_READ acceso compartido de lectura de parte de varios programas
FILE_SHARE_WRITE acceso compartido de escritura de parte de varios programas
FILE_COMMON ubicación del archivo en una carpeta compartida de todos los terminales de cliente \Terminal\Common\Files

delimiter='\t'

[in] valor que se usa como un separador en el archivo txt o csv. Si para el archivo csv el delimitador no está especificado, por defecto se emplea el símbolo de tabulación. Si para el archivo txt el delimitador no está especificado, no se usa ningún separador. Si el valor 0 está establecido como un separador, no se utiliza ningún separador.

codepage=CP_ACP

[in] Valor de la página de código. Están previstas las constantes correspondientes para las páginas de códigos más usadas.

Valor devuelto

En caso de abrir con éxito, la función devuelve el manejador del archivo que luego se usa para acceder a los datos del archivo. En caso de fallo devuelve INVALID_HANDLE.

Nota

© 2000-2025, MetaQuotes Ltd.

Por razones de seguridad el trabajo con los archivos en el lenguaje MQL5 se encuentra bajo un estricto control. Los archivos con los que se realizan las operaciones de archivos utilizando los medios del lenguaje MQL5 no pueden estar fuera del entorno protegido de archivos (file sandbox).

Asegúrese de activar la bandera FILE_ANSI si el archivo debe ser leído en una codificación específica (se indica el parámetro codepage con el valor de la página de código). Si no indica la bandera FILE_ANSI, el archivo de texto se leerá en Unicode sin conversión alguna.

El archivo se abre en la carpeta del terminal de cliente en la subcarpeta MQL5\files (o catálogo_del_agente_de_simulación\MQL5\files en caso de prueba). Si entre las banderas está indicada FILE_COMMON, el archivo se abre en la carpeta compartida de todos los terminales de cliente de MetaTrader5.

"Las tuberías nombradas" pueden ser abiertas de acuerdo con las siguientes reglas:

- El nombre de la tubería es la cadena que debe tener la siguiente apariencia: "\\servername\pipe\pipename", donde servername es el nombre del servidor en la red y pipename es el nombre de la tubería. Si las tuberías se utilizan en el mismo ordenador, se puede omitir el nombre del servidor, pero a su vez hay que poner un punto: "\\.\pipe\pipename". El cliente que intenta conectarse con la tubería tiene que saber su nombre.
- Hay que llamar a las funciones FileFlush() y FileSeek() al inicio del archivo entre las operaciones consecutivas de lectura desde la tubería y la escritura en ella.

En las cadenas mostradas se utiliza un símbolo especial, la barra inversa '\'. Por eso, cuando se escribe el nombre en el programa MQL5, hace falta duplicar la barra inversa '\'. Es decir, escribir el ejemplo arriba mencionado en el código de la siguiente manera: "\\\\servername\\pipe\\pipename".

Puede encontrar más información sobre el trabajo con las tuberías nombradas en el artículo "Communicating With MetaTrader 5 Using Named Pipes Without Using DLLs".

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- modo incorrecto de abrir un archivo
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   string filename=terminal_data_path+"\\MQL5\\Files\\"+"fractals.csv";
   int filehandle=FileOpen(filename,FILE_WRITE|FILE_CSV);
   if(filehandle<0)
     {
      Print("Fallo al abrir el archivo por la ruta absoluta");
      Print("Código de error ",GetLastError());
     }
//--- modo correcto de operar en la zona protegida de archivos
   ResetLastError();
   filehandle=FileOpen("fractals.csv",FILE_WRITE|FILE_CSV);
   if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("FileOpen OK");
     }
   else Print("Operación FileOpen fallida, error ",GetLastError());
//--- otro ejemplo más de crear un directorio anidado dentro MQL5\Files\
   string subfolder="Research";
   filehandle=FileOpen(subfolder+"\\fractals.txt",FILE_WRITE|FILE_CSV);
      if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("El archivo debe ser creado en la carpeta "+terminal_data_path+"\\"+subfol
     }
   else Print("File open failed, error ",GetLastError());
  }
```

Véase también

Uso de página de código, FileFindFirst, FolderCreate, Banderas de apertura de archivos

© 2000-2025, MetaQuotes Ltd.

FileClose

Cierra el archivo previamente abierto por la función FileOpen().

```
void  FileClose(
   int  file_handle      // manejador de archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

No hay valor devuelto.

Ejemplo:

```
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros de entrada
input string InpFileName="file.txt";     // nombre del archivo
input string InpDirectoryName="Data";    // nombre de la carpeta
input int    InpEncodingType=FILE_ANSI; // ANSI=32 o UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprimimos la ruta de la carpeta en la que vamos a trabajar
   PrintFormat("Trabajamos en la carpeta %s\\Files\\",TerminalInfoString(TERMINAL_DATA
//--- reseteamos el valor del error
   ResetLastError();
//--- abrimos el archivo de lectura (si el archivo no existe, se produce un error)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEn
   if(file_handle!=INVALID_HANDLE)
     {
      //--- imprimimos el contenido del archivo
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- cerramos el archivo
      FileClose(file_handle);
     }
   else
      PrintFormat("Error, código = %d",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

FileCopy

Copia el archivo original de una carpeta local o compartida a otro archivo.

```
bool  FileCopy(
   const string  src_file_name,     // nombre del archivo-fuente
   int           common_flag,       // zona de acción
   const string  dst_file_name,     // nombre del archivo de destino
   int           mode_flags         // modo de acceso
   );
```

Parámetros

src_file_name

[in] Nombre del archivo a copiar.

common_flag

[in] Bandera que determina la ubicación del archivo. Si common_flag=FILE_COMMON, entonces el archivo se encuentra en una carpeta compartida de todos los terminales de cliente \Terminal\Common\Files. De lo contrario, el archivo está ubicado en una carpeta local (por ejemplo, *common_flag=0*).

dst_file_name

[in] Nombre del archivo resultante.

mode_flags

[in] Banderas de acceso. El parámetro puede contener sólo 2 banderas: FILE_REWRITE y/o FILE_COMMON - las demás banderas se ignoran. Si el archivo ya existe y la bandera FILE_REWRITE no ha sido especificada, el archivo no se regrabará, y la función devolverá false.

Valor devuelto

En caso de fallo la función devuelve false.

Nota

Si el archivo nuevo ya existía, el copiado va a realizarse dependiendo de la presencia de la bandera FILE_REWRITE en el parámetro mode_flags.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros de entrada
input string InpSrc="source.txt";       // fuente
input string InpDst="destination.txt";  // copia
input int    InpEncodingType=FILE_ANSI; // ANSI=32 o UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- mostramos el contenido de la fuente (tiene que existir)
   if(!FileDisplay(InpSrc))
      return;
//--- comprobamos si existe ya el archivo de la copia (no es obligatorio que haya sido
   if(!FileDisplay(InpDst))
     {
      //--- el archivo de la copia no existe, el copiado sin bandera FILE_REWRITE (cop
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("File is copied!");
      else
         Print("File is not copied!");
     }
   else
     {
      //--- el archivo de la copia ya existe, intentemos copiar sin bandera FILE_REWRI
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("File is copied!");
      else
         Print("File is not copied!");
      //--- el contenido del archivo InpDst se queda el mismo
      FileDisplay(InpDst);
      //--- volvemos a copiar con la bandera FILE_REWRITE (copiado correcto cuando el
      if(FileCopy(InpSrc,0,InpDst,FILE_REWRITE))
         Print("File is copied!");
      else
         Print("File is not copied!");
     }
//--- recibida la copia del archivo InpSrc
   FileDisplay(InpDst);
  }
//+------------------------------------------------------------------+
//| Lectura del contenido de archivo                                 |
//+------------------------------------------------------------------+
bool FileDisplay(const string file_name)
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- abrimos el archivo
   int file_handle=FileOpen(file_name,FILE_READ|FILE_TXT|InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- mostramos en el ciclo el contenido del archivo
      Print("+----------------------+");
      PrintFormat("File name = %s",file_name);
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      Print("+----------------------+");
      //--- cerramos archivo
      FileClose(file_handle);
      return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- no se puede abrir el archivo
   PrintFormat("%s is not opened, error = %d",file_name,GetLastError());
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

FileDelete

Elimina el archivo especificado en una carpeta local del terminal de cliente.

```
bool  FileDelete(
   const string  file_name,     // nombre del archivo a eliminar
   int           common_flag=0  // ubicación del archivo a eliminar
   );
```

Parámetros

file_name

[in] Nombre del archivo.

common_flag=0

[in] Bandera que determina la ubicación del archivo. Si common_flag=FILE_COMMON, entonces el archivo se encuentra en una carpeta compartida de todos los terminales de cliente \Terminal\Common\Files. De lo contrario, el archivo está ubicado en una carpeta local

Valor devuelto

En caso de fallo la función devuelve false.

Nota

Por razones de seguridad el trabajo con los archivos en el lenguaje MQL5 se encuentra bajo un estricto control. Los archivos con los que se realizan las operaciones de archivos utilizando los medios del lenguaje MQL5 no pueden estar fuera del entorno protegido de archivos (file sandbox).

Elimina el archivo especificado en una carpeta local del terminal de cliente (MQL5\Files o MQL5\Tester\Files en caso de prueba). Pero si hay common_flag=FILE_COMMON, la función elimina el archivo de la carpeta compartida de todos los terminales de cliente \Terminal\Common\Files.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- fecha para archivos antiguos
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // variable para almacenar los nombres de archivos
   string   filter="*.txt"; // filtro para la búsqueda de archivos
   datetime create_date;    // fecha de creación del archivo
   string   files[];        // lista con los nombres de los archivos
   int      def_size=25;    // tamaño del array por defecto
   int      size=0;         // número de archivos
//--- adjudicar memoria para array
   ArrayResize(files,def_size);
//--- recibir el manejador de búsqueda en la raíz de la carpeta local
   long search_handle=FileFindFirst(filter,file_name);
//--- comprobamos si la función FileFindFirst() se ha ejecutado con exito
   if(search_handle!=INVALID_HANDLE)
     {
      //--- repasamos los archivos en el ciclo
      do
        {
         files[size]=file_name;
         //--- aumentamos el tamaño del array
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- reseteamos el valor del error
         ResetLastError();
         //--- obtenemos la fecha de creación del archivo
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- comprobamos si el archivo es antiguo
         if(create_date<InpFilesDate)
           {
            PrintFormat("¡El archivo %s ha sido eliminado!",file_name);
            //--- eliminamos el archivo antiguo
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- cerramos el manejador de búsqueda
      FileFindClose(search_handle);
     }
   else
     {
      Print("Files not found!");
      return;
     }
//--- comprobamos qué archivos se han quedado
   PrintFormat("Resultados:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("¡El archivo %s existe!",files[i]);
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("¡El archivo %s ha sido eliminado!",files[i]);
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

FileMove

Mueve un archivo de una carpeta local o compartida a otra carpeta.

```
bool  FileMove(
   const string  src_file_name,     // nombre del archivo para la operación de mover
   int           common_flag,       // zona de acción
   const string  dst_file_name,     // nombre del archivo de destino
   int           mode_flags         // modo de acceso
   );
```

Parámetros

src_file_name

[in] Nombre del archivo a mover/renombrar.

common_flag

[in] Bandera que determina la ubicación del archivo. Si common_flag=FILE_COMMON, entonces el archivo se encuentra en una carpeta compartida de todos los terminales de cliente \Terminal\Common\Files. De lo contrario, el archivo está ubicado en una carpeta local (*common_flag=0*).

dst_file_name

[in] Nombre del archivo resultante.

mode_flags

[in] Banderas de acceso. El parámetro puede contener sólo 2 banderas: FILE_REWRITE y/o FILE_COMMON - las demás banderas se ignoran. Si el archivo ya existe y la bandera FILE_REWRITE no ha sido especificada, el archivo no se regrabará, y la función devolverá false.

Valor devuelto

En caso de fallo la función devuelve false.

Nota

Por razones de seguridad el trabajo con los archivos en el lenguaje MQL5 se encuentra bajo un estricto control. Los archivos con los que se realizan las operaciones de archivos utilizando los medios del lenguaje MQL5 no pueden estar fuera del entorno protegido de archivos (file sandbox).

Si el archivo nuevo ya existía, el copiado va a realizarse dependiendo de la presencia de la bandera FILE_REWRITE en el parámetro mode_flags.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros de entrada
input string InpSrcName="data.txt";
input string InpDstName="newdata.txt";
input string InpSrcDirectory="SomeFolder";
input string InpDstDirectory="OtherFolder";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string local=TerminalInfoString(TERMINAL_DATA_PATH);
   string common=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- obtenemos las rutas de archivos
   string src_path;
   string dst_path;
   StringConcatenate(src_path,InpSrcDirectory,"//",InpSrcName);
   StringConcatenate(dst_path,InpDstDirectory,"//",InpDstName);
//--- comprobamos si existe el archivo de la fuente (si no, a salir)
   if(FileIsExist(src_path))
      PrintFormat("%s file exists in the %s\\Files\\%s folder",InpSrcName,local,InpSrc
   else
     {
      PrintFormat("Error, %s source file not found",InpSrcName);
      return;
     }
//--- comprobamos si existe el archivo del resultado
   if(FileIsExist(dst_path,FILE_COMMON))
     {
      PrintFormat("%s file exists in the %s\\Files\\%s folder",InpDstName,common,InpDs
      //--- el archivo existe, hay que realizar el traslado con la bandera FILE_REWRIT
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON|FILE_REWRITE))
         PrintFormat("%s file moved",InpSrcName);
      else
         PrintFormat("Error! Code = %d",GetLastError());
     }
   else
     {
      PrintFormat("%s file does not exist in the %s\\Files\\%s folder",InpDstName,comm
      //--- el archivo no existe, el traslado se realiza sin la bandera FILE_REWRITE
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON))
         PrintFormat("%s file moved",InpSrcName);
      else
         PrintFormat("Error! Code = %d",GetLastError());
     }
//--- ahora el archivo ya ha sido movido, vamos a comprobarlo
   if(FileIsExist(dst_path,FILE_COMMON) && !FileIsExist(src_path,0))
      Print("Success!");
   else
      Print("Error!");
  }
```

Véase también

FileIsExist

© 2000-2025, MetaQuotes Ltd.

FileFlush

Esta función guarda en el disco todos los datos que se han quedado en el búfer de entrada/salida.

```
void  FileFlush(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

No hay valor devuelto.

Nota

Cuando se ejecuta la operación de escrituro en el archivo, los datos físicos pueden aparecer en él sólo dentro de un período de tiempo. Para que los datos se guarden en el archivo en el acto, hay que utilizar la función FileFlush(). Si esta función no se utiliza, una parte de datos que todavía no han llegado al disco se graban en él forzadamente sólo cuando la función FileClose() cierra el archivo.

Es necesario utilizar esta función cuando los datos que se graban representan un cierto valor. Y hay que tener en cuenta que las llamadas frecuentes a esta función pueden reflejarse en la velocidad del programa.

Hay que llamar a la función FileFlush() entre las operaciones de lectura y escritura de un archivo.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos la ventana de parámetros de entrada al arranciar el script
#property script_show_inputs
//--- nombre de archivo para la escritura
input string InpFileName="example.csv"; // nombre del archivo
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- reseteamos el valor del error
   ResetLastError();
//--- abrimos el archivo
   int file_handle=FileOpen(InpFileName,FILE_READ|FILE_WRITE|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- grabamos los datos en el archivo
      for(int i=0;i<1000;i++)
        {
         //--- llamamos la función de escritura
         FileWrite(file_handle,TimeCurrent(),SymbolInfoDouble(Symbol(),SYMBOL_BID),Sy
         //--- grabamos los datos en el disco con cada 128 iteración
         if((i & 127)==127)
           {
            //--- ahora los datos van a guardarse en el archivo, y en caso de un erro
            FileFlush(file_handle);
            PrintFormat("i = %d, OK",i);
           }
         //--- retraso de 0,01 segundos
         Sleep(10);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
     }
   else
      PrintFormat("Error, código = %d",GetLastError());
  }
```

Véase también

FileClose

© 2000-2025, MetaQuotes Ltd.

FileGetInteger

Recibe una propiedad de números enteros del archivo. Hay 2 variantes de esta función.

1. Obtención de propiedades por el manejador del archivo.

```
long  FileGetInteger(
   int                          file_handle,          // manejador del archivo
   ENUM_FILE_PROPERTY_INTEGER   property_id           // identificador de la propiedad
   );
```

2. Obtención de propiedades por el nombre del archivo.

```
long  FileGetInteger(
   const string                 file_name,            // nombre del archivo
   ENUM_FILE_PROPERTY_INTEGER   property_id,          // identificador de la propiedad
   bool                         common_folder=false   // el archivo se busca en una ca
   );                                                 // o en la carpeta común de todos
```

Parámetros

file_handle

[in] Descriptor de archivos devuelto por la función FileOpen().

file_name

[in] Nombre del archivo.

property_id

[in] Identificador de la propiedad del archivo. El valor puede ser uno de los valores de la enumeración ENUM_FILE_PROPERTY_INTEGER. Si se utiliza la segunda variante de la función, entonces se puede obtener los valores sólo de las siguientes propiedades: FILE_EXISTS, FILE_CREATE_DATE, FILE_MODIFY_DATE, FILE_ACCESS_DATE y FILE_SIZE.

common_folder=false

[in] Indica la ubicación del archivo. Si el parámetro es igual a false, entonces se revisa la carpeta de datos del terminal, en caso contrario se supone que el archivo se encuentra en la carpeta común de todos los terminales de cliente \Terminal\Common\Files (FILE_COMMON).

Valor devuelto

Valor de la propiedad. En caso del error la función devuelve -1. Para obtener el código del error, se debe llamar a la función GetLastError().

Si durante la obtención de propiedades por el nombre se especifica una carpeta, en cualquier caso la función pondrá el error 5018 (ERR_MQL_FILE_IS_DIRECTORY), y el valor devuelto será correcto.

Nota

La función siempre cambia el código del error. En caso de una finalización con éxito el código del error se pone a cero.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos la ventana de parámetros de entrada al arrancar el script
#property script_show_inputs
//--- parámetros de entrada
input string InpFileName="data.csv";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string path=InpDirectoryName+"//"+InpFileName;
   long   l=0;
//--- abrimos el archivo
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_CSV);
   if(handle!=INVALID_HANDLE)
     {
      //--- imprimimos toda la información sobre el archivo
      Print(InpFileName," file info:");
      FileInfo(handle,FILE_EXISTS,l,"bool");
      FileInfo(handle,FILE_CREATE_DATE,l,"date");
      FileInfo(handle,FILE_MODIFY_DATE,l,"date");
      FileInfo(handle,FILE_ACCESS_DATE,l,"date");
      FileInfo(handle,FILE_SIZE,l,"other");
      FileInfo(handle,FILE_POSITION,l,"other");
      FileInfo(handle,FILE_END,l,"bool");
      FileInfo(handle,FILE_IS_COMMON,l,"bool");
      FileInfo(handle,FILE_IS_TEXT,l,"bool");
      FileInfo(handle,FILE_IS_BINARY,l,"bool");
      FileInfo(handle,FILE_IS_CSV,l,"bool");
      FileInfo(handle,FILE_IS_ANSI,l,"bool");
      FileInfo(handle,FILE_IS_READABLE,l,"bool");
      FileInfo(handle,FILE_IS_WRITABLE,l,"bool");
      //--- cerramos el archivo
      FileClose(handle);
     }
   else
      PrintFormat("%s file is not opened, ErrorCode = %d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------+
//| Visualización del valor de la propiedad del archivo
//+------------------------------------------------------------------+
void FileInfo(const int handle,const ENUM_FILE_PROPERTY_INTEGER id,
              long l,const string type)
  {
//--- obtenemos el valor de la propiedad
   ResetLastError();
   if((l=FileGetInteger(handle,id))!=-1)
     {
      //--- valor recibido, vamos a mostrarlo en el formato correcto
      if(!StringCompare(type,"bool"))
         Print(EnumToString(id)," = ",l ? "true" : "false");
      if(!StringCompare(type,"date"))
         Print(EnumToString(id)," = ",(datetime)l);
      if(!StringCompare(type,"other"))
         Print(EnumToString(id)," = ",l);
     }
   else
      Print("Error, Code = ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

Véase también

Operaciones con archivos, Propiedades de archivos

© 2000-2025, MetaQuotes Ltd.

FileIsEnding

Determina el final de un archivo en el proceso de lectura.

```
bool  FileIsEnding(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

La función devuelve true, si se llega el final del archivo durante la lectura o en el proceso de mover el puntero de archivos.

Nota

Para detectar el fin del archivo, la función intenta leer la siguiente cadena del archivo. Si no la hay, la función devuelve true, de lo contrario - false.

Ejemplo:

```
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros de entrada
input string InpFileName="file.txt";     // nombre del archivo
input string InpDirectoryName="Data";    // nombre de la carpeta
input int    InpEncodingType=FILE_ANSI; // ANSI=32 o UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprimimos la ruta de la carpeta en la que vamos a trabajar
   PrintFormat("Trabajamos en la carpeta %s\\Files\\",TerminalInfoString(TERMINAL_DATA
//--- reseteamos el valor del error
   ResetLastError();
//--- abrimos el archivo de lectura (si el archivo no existe, se produce un error)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //--- imprimimos el contenido del archivo
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- cerramos el archivo
      FileClose(file_handle);
     }
   else
      PrintFormat("Error, código = %d",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

FileIsLineEnding

Determina el fin de una línea en un archivo de texto en el proceso de lectura.

```
bool  FileIsLineEnding(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

Devuelve true, si se llega al final de la línea (símbolos CR-LF) durante la lectura de un archivo txt o csv.

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWriteString)

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileIsLineEnding.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- parámetros para la lectura de datos
input string InpFileName="RSI.csv";   // nombre del archivo
input string InpDirectoryName="Data"; // nombre de la carpeta
//--- buferes de indicadores
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- variables de sobrecompra
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- variables de sobreventa
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- variables de tamaños de arrays por defecto
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- adjudicamos memoria para arrays
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- abrimos el archivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Archivo %s abierto para la lectura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PAT
      double value;
      //--- leemos los datos desde el archivo
      while(!FileIsEnding(file_handle))
        {
         //--- leemos el primer valor en la cadena
         value=FileReadNumber(file_handle);
         //--- leemos a diferentes arrays dependiendo del resultado de la función
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
```

© 2000-2025, MetaQuotes Ltd.

```
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos leídos, archivo %s cerrado",InpFileName);
     }
   else
     {
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
      return(INIT_FAILED);
     }
//--- enlace de arrays
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
//---- establecimiento de valores del indicador que no van a ser visibles en el gráfic
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Lectura de datos de la cadena de caracteres del archivo
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- leemos hasta alcanzar el fin de la cadena o archivo
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- desplazamos el carro al leer el número
      if(flag)
         FileReadNumber(file_handle);
      //--- recordamos la fecha actual
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- si hace falta aumentamos el tamaño del array
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- pasamos de la primera iteración
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
{
```

© 2000-2025, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- ciclo para las barras todavía no procesadas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- por defecto 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- prueba de que si hay más datos
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- si las fechas coinciden, la barra se encuentra en la zona de sobrec
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - color rojo
               color_buff[i]=0;
               //--- aumentamos el contador
               ovb_ind=j+1;
               break;
              }
           }
      //--- prueba de que si hay más datos
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- si las fechas coinciden, la barra se encuentra en la zona de sobreve
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - color azul
               color_buff[i]=1;
               //--- aumentamos el contador
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Manejador del evento ChartEvent                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
```

© 2000-2025, MetaQuotes Ltd.

```
                  )
  {
//--- variamos el grosor del indicador en función de la escala
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Véase también

FileWriteString

© 2000-2025, MetaQuotes Ltd.

FileReadArray

Lee los arrays de cualquier tipo, salvo los arrays literales (string) (puede ser un array de estructuras que no contienen las cadenas ni arrays dinámicos) de un archivo binario desde la posición actual del puntero de archivos.

```
uint  FileReadArray(
   int     file_handle,              // manejador del archivo
   void&   array[],                  // array para grabar
   int     start=0,                  // posición de inicio del array para grabar
   int     count=WHOLE_ARRAY         // cantidad para leer
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

array[]

[out] Array donde van a cargarse los datos.

start=0

[in] Posición de inicio para la escritura en el array.

count=WHOLE_ARRAY

[in] Número de elementos para la lectura.

Valor devuelto

Número de elementos leídos. Por defecto lee el array entero (count=WHOLE_ARRAY).

Nota

Un array de cadenas puede ser leído únicamente desde un archivo del tipo TXT. En caso de necesidad la función intenta aumentar el tamaño del array.

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWriteArray)

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos la ventana de parámetros de entrada al arrancar el script
#property script_show_inputs
//--- parámetros de entrada
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Estructura para guardar los datos de precios                     |
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // fecha
   double            bid;  // precio bid
   double            ask;  // precio ask
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- array de estructura
   prices arr[];
//--- ruta del archivo
   string path=InpDirectoryName+"//"+InpFileName;
//--- abrimos archivo
   ResetLastError();
   int file_handle=FileOpen(path,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- leemos todos los datos desde el archivo al array
      FileReadArray(file_handle,arr);
      //--- obtenemos el tamaño del array
      int size=ArraySize(arr);
      //--- imprimimos los datos desde el array
      for(int i=0;i<size;i++)
         Print("Date = ",arr[i].date," Bid = ",arr[i].bid," Ask = ",arr[i].ask);
      Print("Total data = ",size);
      //--- cerramos el archivo
      FileClose(file_handle);
     }
   else
      Print("File open failed, error ",GetLastError());
  }
```

Véase también

Variables, FileWriteArray

© 2000-2025, MetaQuotes Ltd.

FileReadBool

Lee de un archivo del tipo CSV una cadena desde la posición actual hasta el separador (o hasta el final de la cadena de caracteres) y convierte la cadena leída al valor del tipo bool.

```
bool  FileReadBool(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

La cadena leída puede tener valores "true", "false" o una representación simbólica de números enteros "0" o "1". El valor no nulo se convierte al lógico true. La función devuelve el valor convertido que se ha obtenido.

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWrite)

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              Demo_FileReadBool.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- parámetros para la lectura de datos
input string InpFileName="MACD.csv";  // nombre del archivo
input string InpDirectoryName="Data"; // nombre de la carpeta
//--- variables globales
int      ind=0;        // índice
double   upbuff[];     // búfer de indicadores de flechas arriba
double   downbuff[];   // búfer de indicadores de flechas abajo
bool     sign_buff[]; // array de señales (true - compra, false - venta)
datetime time_buff[]; // array de la hora de accionamiento de señales
int      size=0;       // tamaño de arrays de señales
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- abrimos el archivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Archivo %s abierto para la lectura",InpFileName);
      //--- primero leemos el número de señales
      size=(int)FileReadNumber(file_handle);
      //--- adjudicamos la memoria para los arrays
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- leemos los datos desde el archivo
      for(int i=0;i<size;i++)
        {
         //--- tiempo de la señal
         time_buff[i]=FileReadDatetime(file_handle);
         //--- valor de la señal
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
     }
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
       return(INIT_FAILED);
      }
//--- enlace de los arrays
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- establecemos el código del símbolo para dibujar en PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- establecimiento de valores del indicador que no van a ser visibles en el gráfic
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
//--- ciclo para las barras todavía no procesadas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- por defecto 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- prueba de que si hay más datos
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si las fechas coinciden, utilizamos el valor desde el archivo
            if(time[i]==time_buff[j])
              {
               //--- dibujamos la flecha dependiendo de la señal
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- aumentamos el contador
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Tipo bool, FileWrite

© 2000-2025, MetaQuotes Ltd.

FileReadDatetime

Lee de un archivo del tipo CSV una cadena de uno de los formatos: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" o "HH:MI:SS" - y la convierte al valor del tipo datetime.

```
datetime  FileReadDatetime(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

Valor del tipo datetime.

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWrite)

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileReadDateTime.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- parámetros para la lectura de datos
input string InpFileName="MACD.csv";  // nombre del archivo
input string InpDirectoryName="Data"; // nombre de la carpeta
//--- variables globales
int      ind=0;        // índice
double   upbuff[];     // búfer de indicadores de flechas arriba
double   downbuff[];   // búfer de indicadores de flechas abajo
bool     sign_buff[]; // array de señales (true - compra, false - venta)
datetime time_buff[]; // array de la hora de accionamiento de señales
int      size=0;       // tamaño de arrays de señales
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- abrimos el archivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Archivo %s abierto para la lectura",InpFileName);
      //--- primero leemos el número de señales
      size=(int)FileReadNumber(file_handle);
      //--- adjudicamos la memoria para los arrays
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- leemos los datos desde el archivo
      for(int i=0;i<size;i++)
        {
         //--- tiempo de la señal
         time_buff[i]=FileReadDatetime(file_handle);
         //--- valor de la señal
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
     }
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
       return(INIT_FAILED);
      }
//--- enlace de los arrays
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- establecemos el código del símbolo para dibujar en PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- establecimiento de valores del indicador que no van a ser visibles en el gráfic
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
//--- ciclo para las barras todavía no procesadas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- por defecto 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- prueba de que si hay más datos
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si las fechas coinciden, utilizamos el valor desde el archivo
            if(time[i]==time_buff[j])
              {
               //--- dibujamos la flecha dependiendo de la señal
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- aumentamos el contador
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Tipo datetime, StringToTime, TimeToString, FileWrite

© 2000-2025, MetaQuotes Ltd.

FileReadDouble

Lee un número de doble precisión con punto flotante (double) de un archivo binario desde la posición actual del puntero de archivos.

```
double  FileReadDouble(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

Valor del tipo double.

Nota

Para obtener la información sobre el error hay que llamar a la función GetLastError().

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWriteDouble)

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- parámetros para la lectura de datos
input string InpFileName="MA.csv";     // nombre del archivo
input string InpDirectoryName="Data"; // nombre de la carpeta
//--- variables globales
int      ind=0;
int      size=0;
double   ma_buff[];
datetime time_buff[];
//--- indicator buffer
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- abrimos el archivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Archivo %s abierto para la lectura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PAT
      //--- primero leemos cuántos datos contiene el archivo en total
      size=(int)FileReadDouble(file_handle);
      //--- adjudicamos la memoria para los arrays
      ArrayResize(ma_buff,size);
      ArrayResize(time_buff,size);
      //--- leemos los datos desde el archivo
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadDouble(file_handle);
         ma_buff[i]=FileReadDouble(file_handle);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos leídos, archivo %s cerrado",InpFileName);
     }
   else
     {
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
      return(INIT_FAILED);
     }
//--- enlace del array al búfer de indicadores con el índice 0
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- establecimiento de valores del indicador que no van a mostrarse en el gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- ciclo para las barras no procesadas todavía
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- por defecto 0
      buff[i]=0;
      //--- prueba de que si hay más datos
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si las fechas coinciden, utilizamos el valor desde el archivo
            if(time[i]==time_buff[j])
              {
               buff[i]=ma_buff[j];
               //--- aumentamos el contador
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Tipos reales (double, float), StringToDouble, DoubleToString, FileWriteDouble

© 2000-2025, MetaQuotes Ltd.

FileReadFloat

Lee el número de precisión simple en punto flotante (float) de un archivo binario desde la posición actual.

```
float  FileReadFloat(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la funciónFileOpen().

Valor devuelto

Valor del tipo float.

Nota

Para obtener la información sobre el error hay que llamar a la función GetLastError().

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWriteFloat)

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                            Demo_FileReadFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "CloseLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- parámetros para la lectura de datos
input string InpFileName="Close.bin"; // nombre del archivo
input string InpDirectoryName="Data"; // nombre de la carpeta
//--- variables globales
int      ind=0;
int      size=0;
double   close_buff[];
datetime time_buff[];
//--- indicator buffers
double   buff[];
double   color_buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- adjudicamos la memoria para los arrays
   ArrayResize(close_buff,def_size);
   ArrayResize(time_buff,def_size);
//--- abrimos el archivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Archivo %s abierto para la lectura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PAT
      //--- leemos los datos desde el archivo
      while(!FileIsEnding(file_handle))
        {
         //--- leemos los valores de la hora y el precio
         time_buff[size]=(datetime)FileReadDouble(file_handle);
         close_buff[size]=(double)FileReadFloat(file_handle);
         size++;
         //--- aumentamos el tamaño de los arrays si están sobrecargados
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(close_buff,def_size);
            ArrayResize(time_buff,def_size);
           }
        }
      //--- cerramos el archivo
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Datos leídos, archivo %s cerrado",InpFileName);
     }
   else
     {
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
      return(INIT_FAILED);
     }
//--- enlace de los arrays a los búferes de indicadores
   SetIndexBuffer(0,buff,INDICATOR_DATA);
   SetIndexBuffer(1,color_buff,INDICATOR_COLOR_INDEX);
//---- establecimiento de valores del indicador que no van a mostrarse en el gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- ciclo para las barras no procesadas todavía
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- por defecto 0
      buff[i]=0;
      color_buff[i]=0; // el color rojo por defecto
      //--- prueba de que si hay más datos
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si las fechas coinciden, utilizamos el valor desde el archivo
            if(time[i]==time_buff[j])
              {
               //--- obtenemos el precio
               buff[i]=close_buff[j];
               //--- si el precio actual supera el anterior, el color es azul
               if(buff[i-1]>buff[i])
                  color_buff[i]=1;
               //--- aumentamos el contador
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

© 2000-2025, MetaQuotes Ltd.

Tipos reales (double, float), FileReadDouble, FileWriteFloat

© 2000-2025, MetaQuotes Ltd.

FileReadInteger

La función lee de un archivo binario el valor del tipo int, short o char dependiendo de la longitud indicada en bytes. La lectura se realiza desde la posición actual del puntero de archivos.

```
int  FileReadInteger(
   int  file_handle,         // manejador del archivo
   int  size=INT_VALUE       // tamaño del tipo entero en bytes
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

size=INT_VALUE

[in] Número de bytes (hasta 4 inclusive) que hay que leer. Están previstas las siguientes constantes: CHAR_VALUE=1, SHORT_VALUE=2 y INT_VALUE=4, así la función puede leer el valor entero del tipo char, short o int.

Valor devuelto

Valor del tipo int. Es necesario convertir explicitamente el resultado de esta función a un tipo fuente, es decir a aquel tipo de datos que hace falta leer. Puesto que se devuelve el valor del tipo int, se puede convertirlo tranquilamente a cualquier valor entero. El puntero de archivo se desplaza a la cantidad de bytes leídos.

Nota

Cuando se lee menos de 4 bytes, el resultado obtenido será siempre positivo. Si se lee uno o dos bytes, se puede determinar exactamente el signo del número mediante la conversión explícita al tipo char (1 byte) o al tipo short (2 bytes), respectivamente. La obtención del signo para un número de tres bytes no es trivial, ya que no hay tipo base correspondiente.

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWriteInteger)

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Trends"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parámetros para la lectura de datos
input string InpFileName="Trend.bin"; // nombre del archivo
input string InpDirectoryName="Data"; // nombre de la carpeta
//--- variables globales
int      ind=0;
int      size=0;
datetime time_buff[];
//--- indicator buffers
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- adjudicamos la memoria para el array
   ArrayResize(time_buff,def_size);
//--- abrimos el archivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Archivo %s abierto para la lectura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PAT
      //--- variables auxiliares
      int    arr_size;
      uchar  arr[];
      //--- leemos los datos desde el archivo
      while(!FileIsEnding(file_handle))
        {
         //--- averiguamos cuántos símbolos han sido utilizados para la escritura del
         arr_size=FileReadInteger(file_handle,INT_VALUE);
         ArrayResize(arr,arr_size);
         for(int i=0;i<arr_size;i++)
            arr[i]=(char)FileReadInteger(file_handle,CHAR_VALUE);
         //--- recordamos el valor del tiempo
         time_buff[size]=StringToTime(CharArrayToString(arr));
         size++;
         //--- aumentamos el tamaño de los arrays si están sobrecargados
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(time_buff,def_size);
           }
```

© 2000-2025, MetaQuotes Ltd.

```
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos leídos, archivo %s cerrado",InpFileName);
     }
   else
     {
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
      return(INIT_FAILED);
     }
//--- enlace del array al búfer de indicadores
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- establecimiento de valores del indicador que no van a mostrarse en el gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(close,false);
//--- ciclo para las barras no procesadas todavía
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- por defecto 0
      buff[i]=0;
      //--- prueba de que si hay más datos
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si las fechas coinciden, utilizamos el valor desde el archivo
            if(time[i]==time_buff[j])
              {
               //--- obtenemos el precio
               buff[i]=close[i];
               //--- aumentamos el contador
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

© 2000-2025, MetaQuotes Ltd.

IntegerToString, StringToInteger, Tipos enteros, FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileReadLong

Lee el número entero del tipo long (8 bytes) desde la posición actual del puntero de archivos de un archivo binario.

```
long  FileReadLong(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

Valor del tipo long.

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWriteLong)

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              Demo_FileReadLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Volume"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrYellow
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- parámetros para la lectura de datos
input string InpFileName="Volume.bin"; // nombre del archivo
input string InpDirectoryName="Data";  // nombre de la carpeta
//--- variables globales
int      ind=0;
int      size=0;
long     volume_buff[];
datetime time_buff[];
//--- indicator buffers
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- abrimos el archivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Archivo %s abierto para la escritura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PAT
      //--- primero leemos cuántos datos contiene el archivo en total
      size=(int)FileReadLong(file_handle);
      //--- adjudicamos la memoria para los arrays
      ArrayResize(volume_buff,size);
      ArrayResize(time_buff,size);
      //--- leemos los datos desde el archivo
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadLong(file_handle);
         volume_buff[i]=FileReadLong(file_handle);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos leídos, archivo %s cerrado",InpFileName);
     }
   else
     {
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
      return(INIT_FAILED);
     }
//--- enlace del array al búfer de indicadores con el índice 0
   SetIndexBuffer(0,buff,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
//---- establecimiento de valores del indicador que no van a mostrarse en el gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- ciclo para las barras no procesadas todavía
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- por defecto 0
      buff[i]=0;
      //--- prueba de que si hay más datos
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si las fechas coinciden, utilizamos el valor desde el archivo
            if(time[i]==time_buff[j])
              {
               buff[i]=(double)volume_buff[j];
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Tipos enteros, FileReadInteger, FileWriteLong

© 2000-2025, MetaQuotes Ltd.

FileReadNumber

Lee de un archivo del tipo CSV una cadena desde la posición actual hasta el separador (o hasta el final de la línea de texto) y convierte la cadena leída al valor del tipo double.

```
double  FileReadNumber(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

Valor del tipo double.

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWriteString)

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadNumber.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- parámetros para la lectura de datos
input string InpFileName="RSI.csv";   // nombre del archivo
input string InpDirectoryName="Data"; // nombre de la carpeta
//--- buferes de indicadores
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- variables de sobrecompra
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- variables de sobreventa
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- variables de tamaños de arrays por defecto
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- adjudicamos memoria para arrays
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- abrimos el archivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Archivo %s abierto para la lectura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PAT
      double value;
      //--- leemos los datos desde el archivo
      while(!FileIsEnding(file_handle))
        {
         //--- leemos el primer valor en la cadena
         value=FileReadNumber(file_handle);
         //--- leemos a diferentes arrays dependiendo del resultado de la función
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
```

© 2000-2025, MetaQuotes Ltd.

```
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos leídos, archivo %s cerrado",InpFileName);
     }
   else
     {
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
      return(INIT_FAILED);
     }
//--- enlace de arrays
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
//---- establecimiento de valores del indicador que no van a ser visibles en el gráfic
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Lectura de datos de la cadena de caracteres del archivo
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- leemos hasta alcanzar el fin de la cadena o archivo
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- desplazamos el carro al leer el número
      if(flag)
         FileReadNumber(file_handle);
      //--- recordamos la fecha actual
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- si hace falta aumentamos el tamaño del array
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- pasamos de la primera iteración
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
{
```

© 2000-2025, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- ciclo para las barras todavía no procesadas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- por defecto 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- prueba de que si hay más datos
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- si las fechas coinciden, la barra se encuentra en la zona de sobrec
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - color rojo
               color_buff[i]=0;
               //--- aumentamos el contador
               ovb_ind=j+1;
               break;
              }
           }
      //--- prueba de que si hay más datos
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- si las fechas coinciden, la barra se encuentra en la zona de sobreve
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - color azul
               color_buff[i]=1;
               //--- aumentamos el contador
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Manejador del evento ChartEvent                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
```

© 2000-2025, MetaQuotes Ltd.

```
                  )
  {
//--- variamos el grosor del indicador en función de la escala
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Véase también

FileWriteString

© 2000-2025, MetaQuotes Ltd.

FileReadString

Lee del archivo una cadena desde la posición actual del puntero de archivos.

```
string  FileReadString(
   int  file_handle,     // manejador del archivo
   int  length=-1        // longitud de la cadena
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

length=-1

[in] Número de caracteres para leer.

Valor devuelto

Cadena leída (string).

Nota

Cuando leemos de un archivo bin es obligatorio indicar la longitud de la cadena. Cuando leemos de un archivo txt no hace falta especificar la longitud de la cadena; la cadena será leída desde la posición actual hasta el salto de línea "\r\n". Cuando leemos de un archivo csv tampoco hace falta especificar la longitud de la cadena; la cadena será leída desde la posición actual hasta el separador más cercano o hasta el salto de línea.

Si el archivo está abierto con la bandera FILE_ANSI, la cadena leída se convierte a Unicode.

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWriteInteger)

© 2000-2025, MetaQuotes Ltd.

```
///--- mostramos la ventana de los parámetros de entrada al arrancar el script
#property script_show_inputs
//--- parámetros para la lectura de datos
input string InpFileName="Trend.bin"; // nombre del archivo
input string InpDirectoryName="Data"; // nombre de la carpeta
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- abrimos el archivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("El archivo %s está abierto para la lectura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PAT
      //--- variables auxiliares
      int    str_size;
      string str;
      //--- leemos los datos desde el archivo
      while(!FileIsEnding(file_handle))
        {
         //--- averiguamos cuántos símbolos han sido utilizados para la escritura del
         str_size=FileReadInteger(file_handle,INT_VALUE);
         //--- leemos la cadena
         str=FileReadString(file_handle,str_size);
         //--- imprimimos la cadena
         PrintFormat(str);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos leídos, archivo %s cerrado",InpFileName);
     }
   else
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
  }
```

Véase también

Tipo string, Conversión de datos, FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileReadStruct

Lee de un archivo binario el contenido en una estructura que ha sido pasada como un parámetro. La lectura se realiza desde la posición actual del puntero de archivos.

```
uint  FileReadStruct(
   int           file_handle,        // manejador del archivo
   const void&   struct_object,      // estructura de destino para la lectura
   int           size=-1             // tamaño de estructura en bytes
   );
```

Parámetros

file_handle

[in] Descriptor de archivos de un archivo binario abierto.

struct_object

[out] Objeto de esta estructura. La estructura no debe contener líneas de caracteres, matrices dinámicas, funciones virtuales u objetos de clases, así como punteros a objetos y funciones.

size=-1

[in] Cantidad de bytes que hay que leer. Si el tamaño no se especifica o la cantidad de bytes especificada supera el tamaño de la estructura, entonces se usa el tamaño exacto de la estructura indicada.

Valor devuelto

En caso de éxito la función devuelve el número de bytes leído. El puntero de archivos se mueve a esta misma cantidad de bytes.

Ejemplo (se utiliza el archivo conseguido como resultado de trabajo del ejemplo para la función FileWriteStruct)

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Candles"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
#property indicator_separate_window
//--- parámetros para recibir datos
input string  InpFileName="EURUSD.txt"; // nombre del archivo
input string  InpDirectoryName="Data";  // nombre de la carpeta
//+------------------------------------------------------------------+
//| Estructura para almacenar los datos de la vela                   |
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // precio de apertura
   double            close; // precio de cierre
   double            high;  // precio máximo
   double            low;   // precio mínimo
   datetime          date;  // fecha
  };
//--- búferes de indicadores
double       open_buff[];
double       close_buff[];
double       high_buff[];
double       low_buff[];
//--- variables globales
candlesticks cand_buff[];
int          size=0;
int          ind=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int default_size=100;
   ArrayResize(cand_buff,default_size);
//--- abrimos el archivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Archivo %s abierto para la lectura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDA
      //--- leemos los datos desde el archivo
      while(!FileIsEnding(file_handle))
        {
         //--- escribimos los datos en el array
         FileReadStruct(file_handle,cand_buff[size]);
         size++;
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- comprobamos si el array está sobrecargado
         if(size==default_size)
           {
            //--- aumentar la dimensión del array
            default_size+=100;
            ArrayResize(cand_buff,default_size);
           }
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos leídos, archivo %s cerrado",InpFileName);
     }
   else
     {
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
      return(INIT_FAILED);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
//--- valor vacío
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
//--- ciclo para las velas no procesadas todavía
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- por defecto 0
      open_buff[i]=0;
      close_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      //--- prueba de que si hay más datos
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si las fechas coinciden, utilizamos el valor desde el archivo
            if(time[i]==cand_buff[j].date)
              {
               open_buff[i]=cand_buff[j].open;
               close_buff[i]=cand_buff[j].close;
               high_buff[i]=cand_buff[j].high;
               low_buff[i]=cand_buff[j].low;
               //--- aumentamos el contador
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Estructuras y clases, FileWriteStruct

© 2000-2025, MetaQuotes Ltd.

FileSeek

Mueve la posición del puntero de archivos a una cantidad de bytes especificada respecto a la posición indicada.

```
bool  FileSeek(
   int                  file_handle,     // manejador del archivo
   long                 offset,          // en bytes
   ENUM_FILE_POSITION   origin           // posición de referencia
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

offset

[in] Desplazamiento en bytes (puede también adquirir un valor negativo).

origin

[in] Punto de referencia para el desplazamiento. Puede ser uno de los valores de la enumeración ENUM_FILE_POSITION.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false. Para obtener la información sobre el error hay que llamar a la función GetLastError().

Nota

Si el resultado de ejecución de la función FileSeek() es un desplazamiento negativo (sobrepasando "el límite izquierdo" del archivo), el puntero de archivos será puesto en el principio del archivo.

Si la posición se pone fuera del "límite derecho" del archivo (más que el tamaño del archivo), la siguiente escritura en el archivo será realizada no desde el final sino desde la posición puesta. En este caso entre el final anterior del archivo y posición puesta habrán unos valores indeterminados.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSeek.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros de entrada
input string InpFileName="file.txt";     // nombre del archivo
input string InpDirectoryName="Data";    // nombre de la carpeta
input int    InpEncodingType=FILE_ANSI; // ANSI=32 o UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- especificamos el valor de la variable para generar los números aleatorios
   _RandomSeed=GetTickCount();
//--- variable para las posiciones de inicio de cadenas
   ulong pos[];
   int   size;
//--- reseteamos el valor del error
   ResetLastError();
//--- abrimos el archivo
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("El archivo %s está abierto para la lectura",InpFileName);
      //--- obtenemos la posición de inicio para cada cadena de caracteres en el archi
      GetStringPositions(file_handle,pos);
      //--- averiguamos el número total de cadenas en el archivo
      size=ArraySize(pos);
      if(!size)
        {
         //--- si en el archivo no hay cadenas, finalizamos el trabajo
         PrintFormat("¡El archivo %s esta vacío!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- elegimos el número de una cadena al azar
      int ind=MathRand()%size;
      //--- desplazamos la posición al inicio de esta cadena
      if(FileSeek(file_handle,pos[ind],SEEK_SET)==true)
        {
      //--- leemos e imprimimos la cadena con el número ind
         PrintFormat("String text with %d number: \"%s\"",ind,FileReadString(file_hanc
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("El archivo %s ha sido cerrado",InpFileName);
     }
   else
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
  }
//+------------------------------------------------------------------+
//| La función determina las posiciones de inicio para cada una de las cadenas en el a
//| las coloca en el array arr                                       |
//+------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- tamaño del array por defecto
   int def_size=127;
//--- adjudicamos la memoria para el array
   ArrayResize(arr,def_size);
//--- contador de cadenas
   int i=0;
//--- si no es el fin del archivo, entonces por lo menos hay una cadena
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // el archivo está vacío, salimos
//--- determinamos el desplazamiento en bytes dependiendo de la codificación
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- repasamos las cadenas en el ciclo
   while(1)
     {
      //--- leemos la cadena
      FileReadString(handle);
      //--- prueba en el fin del archivo
      if(!FileIsEnding(handle))
        {
         //--- recordamos la posición de la siguiente cadena
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- aumentamos el tamaño del array si está sobrecargado
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // fin del archivo, salimos
     }
//--- definimos el tamaño real del array
   ArrayResize(arr,i);
  }
```

© 2000-2025, MetaQuotes Ltd.

FileSize

Devuelve el tamaño del archivo en bytes.

```
ulong  FileSize(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

Valor del tipo int.

Nota

Para obtener la información sobre el error hay que llamar a la función GetLastError().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros de entrada
input ulong  InpThresholdSize=20;        // umbral del tamaño de archivos en kilobytes
input string InpBigFolderName="big";     // carpeta para archivos grandes
input string InpSmallFolderName="small"; // carpeta para archivos pequeños
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // variable para almacenar los nombres de archivos
   string   filter="*.csv"; // filtro para buscar archivos
   ulong    file_size=0;    // tamaño del archivo en bytes
   int      size=0;         // número de archivos
//--- imprimimos la ruta de la carpeta en la que vamos a trabajar
   PrintFormat("Trabajamos en la carpeta %s\\Files\\",TerminalInfoString(TERMINAL_COMM
//--- recepción del manejador de búsqueda en la raíz de la carpeta común de todos los
   long search_handle=FileFindFirst(filter,file_name,FILE_COMMON);
//--- comprobamos si la función FileFindFirst()ha terminado su trabajo con éxito
   if(search_handle!=INVALID_HANDLE)
     {
      //--- movemos los archivos en el ciclo dependiendo de su tamaño
      do
        {
         //--- abrimos el archivo
         ResetLastError();
         int file_handle=FileOpen(file_name,FILE_READ|FILE_CSV|FILE_COMMON);
         if(file_handle!=INVALID_HANDLE)
           {
            //--- obtenemos el tamaño del archivo
            file_size=FileSize(file_handle);
            //--- cerramos el archivo
            FileClose(file_handle);
           }
         else
           {
            PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",file_nar
            continue;
           }
         //--- imprimimos el tamaño del archivo
         PrintFormat("El tamaño del archivo %s es de %d bytes",file_name,file_size);
         //--- definimos la ruta para mover el archivo
         string path;
         if(file_size>InpThresholdSize*1024)
            path=InpBigFolderName+"//"+file_name;
         else
            path=InpSmallFolderName+"//"+file_name;
         //--- desplazamos el archivo
         ResetLastError();
         if(FileMove(file_name,FILE_COMMON,path,FILE_REWRITE|FILE_COMMON))
            PrintFormat("El archivo %s ha sido desplazado",file_name);
         else
            PrintFormat("Error, código = %d",GetLastError());
        }
      while(FileFindNext(search_handle,file_name));
      //--- cerramos el manejador de búsqueda
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

FileTell

Devuelve la posición actual del puntero de archivos de un archivo abierto correspondiente.

```
ulong  FileTell(
   int  file_handle      // manejador del archivo
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

Valor devuelto

Posición actual del descriptor de archivos desde el principio del archivo.

Nota

Para obtener la información sobre el error hay que llamar a la función GetLastError().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                Demo_FileTell.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros de entrada
input string InpFileName="file.txt";     // nombre del archivo
input string InpDirectoryName="Data";    // nombre de la carpeta
input int    InpEncodingType=FILE_ANSI; // ANSI=32 o UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- especificamos el valor de la variable para generar los números aleatorios
   _RandomSeed=GetTickCount();
//--- variable para las posiciones de inicio de cadenas
   ulong pos[];
   int   size;
//--- reseteamos el valor del error
   ResetLastError();
//--- abrimos el archivo
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("El archivo %s está abierto para la lectura",InpFileName);
      //--- obtenemos la posición de inicio para cada cadena de caracteres en el archi
      GetStringPositions(file_handle,pos);
      //--- averiguamos el número total de cadenas en el archivo
      size=ArraySize(pos);
      if(!size)
        {
         //--- si en el archivo no hay cadenas, finalizamos el trabajo
         PrintFormat("¡El archivo %s esta vacío!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- elegimos el número de una cadena al azar
      int ind=MathRand()%size;
      //--- desplazamos la posición al inicio de esta cadena
      FileSeek(file_handle,pos[ind],SEEK_SET);
      //--- leemos e imprimimos la cadena con el número ind
      PrintFormat("Texto de la cadena con el número %d: \"%s\"",ind,FileReadString(fil
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("El archivo %s ha sido cerrado",InpFileName);
     }
   else
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
  }
//+------------------------------------------------------------------+
//| La función determina las posiciones de inicio para cada una de las cadenas en el a
//| las coloca en el array arr                                       |
//+------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- tamaño del array por defecto
   int def_size=127;
//--- adjudicamos la memoria para el array
   ArrayResize(arr,def_size);
//--- contador de cadenas
   int i=0;
//--- si no es el fin del archivo, entonces por lo menos hay una cadena
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // el archivo está vacío, salimos
//--- determinamos el desplazamiento en bytes dependiendo de la codificación
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- repasamos las cadenas en el ciclo
   while(1)
     {
      //--- leemos la cadena
      FileReadString(handle);
      //--- prueba en el fin del archivo
      if(!FileIsEnding(handle))
        {
         //--- recordamos la posición de la siguiente cadena
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- aumentamos el tamaño del array si está sobrecargado
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // fin del archivo, salimos
     }
//--- definimos el tamaño real del array
   ArrayResize(arr,i);
  }
```

© 2000-2025, MetaQuotes Ltd.

FileWrite

Esta función se utiliza para escribir los datos en un archivo del tipo CSV o TXT. Si el delimitador no es igual a 0, entonces se inserta entre los datos automáticamente. Después de la escritura en el archivo, se añade el salto de línea "\r\n".

```
uint  FileWrite(
   int  file_handle,      // manejador del archivo
   ...                    // lista de parámetros a escribir
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

...

[in] Lista de parámetros separados por coma para escribirlos en el archivo. El número de parámetros a introducir no puede superar 63.

Valor devuelto

Número de bytes escritos.

Nota

En la salida los datos numéricos se convierten al formato de texto (ver la función Print()). Los datos del tipo double se visualizan con la precisión de hasta 16 dígitos decimales después del punto, además, los datos pueden ser visualizados en el formato tradicional o científico, dependiendo de cuál de ellos va a ser más compacto. Los datos del tipo float se visualizan con 5 dígitos decimales después del punto. Para visualizar los números reales con otra precisión o en un formato explícitamente especificado hay que usar la función DoubleToString().

Los números del tipo bool se visualizan como cadenas "true" o "false". Los números del tipo datetime se visualizan en el formato "YYYY.MM.DD HH:MI:SS".

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                Demo_FileWrite.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros para recibir datos desde el terminal
input string             InpSymbolName="EURUSD";           // par de divisas
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // período de tiempo
input int                InpFastEMAPeriod=12;              // período de la EMA rápida
input int                InpSlowEMAPeriod=26;              // período de la EMA lenta
input int                InpSignalPeriod=9;                // período del promedio de
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // tipo del precio
input datetime           InpDateStart=D'2012.01.01 00:00'; // fecha de inicio del copi
//--- parámetros para la escritura de datos en el archivo
input string             InpFileName="MACD.csv";  // nombre del archivo
input string             InpDirectoryName="Data"; // nombre de la carpeta
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // fecha del fin de copiado de datos
   bool     sign_buff[]; // array de señales (true - compra, false - venta)
   datetime time_buff[]; // array del tiempo de llegada de señales
   int      sign_size=0; // tamaño de arrays de las señales
   double   macd_buff[]; // array de los valores del indicador
   datetime date_buff[]; // array de las fechas del indicador
   int      macd_size=0; // tamaño de arrays del indicador
//--- tiempo de finalización - actual
   date_finish=TimeCurrent();
//--- obtenemos el manejador del indicador MACD
   ResetLastError();
   int macd_handle=iMACD(InpSymbolName,InpSymbolPeriod,InpFastEMAPeriod,InpSlowEMAPeri
   if(macd_handle==INVALID_HANDLE)
     {
      //--- fallo al obtener el manejador del indicador
      PrintFormat("Error a la hora de obtener el manejador del indicador. Código del e
      return;
     }
//--- seguimos en el ciclo hasta que el indicador no termine de calcular todos sus val
   while(BarsCalculated(macd_handle)==-1)
      Sleep(10); // retardo para que al indicador le de tiempo a calcular sus valores
//--- copiamos los valores del indicador durante un período determinado
   ResetLastError();
   if(CopyBuffer(macd_handle,0,InpDateStart,date_finish,macd_buff)==-1)
     {
      PrintFormat("Fallo al copiar los valores del indicador. Código del error = %d",G
      return;
     }
//--- copiamos el tiempo correspondiente para los valores del indicador
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("Fallo al copiar los valores del tiempo. Código del error = %d",GetL
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- liberamos la memoria que ocupa el indicador
   IndicatorRelease(macd_handle);
//--- obtenemos el tamaño del búfer
   macd_size=ArraySize(macd_buff);
//--- vamos a analizar los datos y guardar las señales del indicador en los arrays
   ArrayResize(sign_buff,macd_size-1);
   ArrayResize(time_buff,macd_size-1);
   for(int i=1;i<macd_size;i++)
     {
      //--- señales de compra
      if(macd_buff[i-1]<0 && macd_buff[i]>=0)
        {
         sign_buff[sign_size]=true;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
      //--- señales de venta
      if(macd_buff[i-1]>0 && macd_buff[i]<=0)
        {
         sign_buff[sign_size]=false;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
     }
//--- abrimos el archivo para escribir los valores del indicador (si no existe, se cre
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("El archivo %s está abierto para la escritura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PA
      //--- primero apuntamos el número de señales
      FileWrite(file_handle,sign_size);
      //--- apuntamos en el archivo el tiempo de señales y sus valores
      for(int i=0;i<sign_size;i++)
         FileWrite(file_handle,time_buff[i],sign_buff[i]);
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos grabados, el archivo %s cerrado",InpFileName);
     }
   else
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
  }
```

Véase también

Comment, Print, StringFormat

© 2000-2025, MetaQuotes Ltd.

FileWriteArray

Esta función escribe los arrays de cualquier tipo, excepto los arrays para las cadenas, en un archivo del tipo BIN (puede ser un array de estructuras que no contienen cadenas ni arrays dinámicos).

```
uint  FileWriteArray(
   int           file_handle,          // manejador del archivo
   const void&   array[],              // matriz
   int           start=0,              // índice inicial en el array
   int           count=WHOLE_ARRAY     // número de elementos
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

array[]

[out] Array a grabar.

start=0

[in] Índice inicial en el array (número del primer elemento a grabar).

count=WHOLE_ARRAY

[in] Número de elementos que se graban (WHOLE_ARRAY significa que se graban todos los elementos empezando desde el número start hasta el final del array).

Valor devuelto

Número de elementos grabados.

Nota

Un array de cadenas puede ser grabado sólo en un archivo del tipo TXT. En este caso las cadenas automáticamente se acaban con los caracteres de salto de línea "\r\n". Dependiendo del tipo de archivo ANSI o UNICODE, las cadenas se convierten a la codificación ansi o no.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteArray.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- parámetros de entrada
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Estructura para almacenar los datos sobre los precios
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // fecha
   double            bid;  // precio bid
   double            ask;  // precio ask
  };
//--- variables globales
int    count=0;
int    size=20;
string path=InpDirectoryName+"//"+InpFileName;
prices arr[];
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- adjudicación de la memoria para el array
   ArrayResize(arr,size);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- escribir count cadenas restantes, si count<n
   WriteData(count);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- guardamos los datos en el array
   arr[count].date=TimeCurrent();
   arr[count].bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   arr[count].ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- mostramos los datos actuales
   Print("Date = ",arr[count].date," Bid = ",arr[count].bid," Ask = ",arr[count].ask);
//--- aumentamos el contador
   count++;
//--- si el array ya esta lleno, escribimos los datos en el archivo y lo ponemos a cer
   if(count==size)
     {
      WriteData(size);
      count=0;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
//+------------------------------------------------------------------+
//| Escritura de n elementos del array en el archivo                 |
//+------------------------------------------------------------------+
void WriteData(const int n)
   {
//--- abrimos el archivo
    ResetLastError();
    int handle=FileOpen(path,FILE_READ|FILE_WRITE|FILE_BIN);
    if(handle!=INVALID_HANDLE)
      {
       //--- escribimos los datos del array al final del archivo
       FileSeek(handle,0,SEEK_END);
       FileWriteArray(handle,arr,0,n);
       //--- cerramos el archivo
       FileClose(handle);
      }
    else
       Print("Failed to open the file, error ",GetLastError());
   }
```

Véase también

Variables, FileSeek

© 2000-2025, MetaQuotes Ltd.

FileWriteDouble

Escribe el valor del parámetro del tipo double desde la posición actual del puntero de archivos en un archivo binario.

```
uint  FileWriteDouble(
   int     file_handle,     // manejador del archivo
   double  value            // valor para escribir
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

value

[in] Valor del tipo double.

Valor devuelto

En caso del éxito la función devuelve el número de bytes escritos (en este caso sizeof(double)=8). El puntero de archivos se mueve a esta misma cantidad de bytes.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros para obtener datos desde el terminal
input string             InpSymbolName="EURJPY";           // par de divisas
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_M15;       // período de tiempo
input int                InpMAPeriod=10;                   // período de suavizado
input int                InpMAShift=0;                     // desplazamiento del indic
input ENUM_MA_METHOD     InpMAMethod=MODE_SMA;             // tipo de suavizado
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // tipo del precio
input datetime           InpDateStart=D'2013.01.01 00:00'; // fecha de inicio del copi
//--- parámetros para la escritura de datos en el archivo
input string             InpFileName="MA.csv";    // nombre del archivo
input string             InpDirectoryName="Data"; // nombre de la carpeta
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   ma_buff[];
   datetime time_buff[];
   int      size;
//--- obtenemos el indicador del manejador MA
   ResetLastError();
   int ma_handle=iMA(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpMAShift,InpMAMethod,
   if(ma_handle==INVALID_HANDLE)
     {
      //--- fallo al obtener el manejador del indicador
      PrintFormat("Fallo al obtener el manejador del indicador. Código del error = %d"
      return;
     }
//--- seguimos en el ciclo hasta que el indicador no termine de calcular todos sus val
   while(BarsCalculated(ma_handle)==-1)
      Sleep(20); // retardo para que al indicador le de tiempo a calcular sus valores
   PrintFormat("En el archivo serán escritos los valores del indicador a partir del %s
//--- copiamos los valpres del indicador
   ResetLastError();
   if(CopyBuffer(ma_handle,0,InpDateStart,date_finish,ma_buff)==-1)
     {
      PrintFormat("Fallo al copiar los valores del indicador. Código del error = %d",G
      return;
     }
//--- copiamos el tiempo de aparición de barras correspondientes
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Fallo al copiar los valores del tiempo. Código del error = %d",GetL
      return;
     }
//--- obtenemos el tamaño del búfer
   size=ArraySize(ma_buff);
//--- liberamos la memoria que ocupa el indicador
   IndicatorRelease(ma_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- abrimos el archivo para escribir los valores del indicador (si no existe, se cre
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("El archivo %s está abierto para la escritura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PAT
      //--- primero escribimos el tamaño de la muestra de datos
      FileWriteDouble(file_handle,(double)size);
      //--- escribimos en el archivo el tiempo y los valores del indicador
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteDouble(file_handle,ma_buff[i]);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos grabados, el archivo %s cerrado",InpFileName);
     }
   else
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
  }
```

Véase también

Tipos reales (double, float)

© 2000-2025, MetaQuotes Ltd.

FileWriteFloat

Escribe el valor del parámetro del tipo float desde la posición actual del puntero de archivos en un archivo binario.

```
uint  FileWriteFloat(
   int    file_handle,     // manejador del archivo
   float  value            // valor para escribir
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

value

[in] Valor del tipo float.

Valor devuelto

En caso del éxito la función devuelve el número de bytes escritos (en este caso sizeof(float)=4). El puntero de archivos se mueve a esta misma cantidad de bytes.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros para obtener datos desde el terminal
input string          InpSymbolName="EURUSD";           // par de divisas
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_M15;       // período de tiempo
input datetime        InpDateStart=D'2013.01.01 00:00'; // fecha de inicio del copiad
//--- parámetros para la escritura de datos en el archivo
input string          InpFileName="Close.bin"; // nombre del archivo
input string          InpDirectoryName="Data"; // nombre de la carpeta
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- reseteamos el valor del error
   ResetLastError();
//--- copiamos el precio de cierre para cada barra
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Fallo al copiar los valores de los precios de cierre. Código del er
      return;
     }
//--- copiamos el tiempo para cada barra
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Fallo al copiar el valor del tiempo. Código del error = %d",GetLast
      return;
     }
//--- obtenemos el tamaño del búfer
   size=ArraySize(close_buff);
//--- abrimos el archivo para escribir los valores (si no existe, se crea automáticame
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("El archivo %s está abierto para la escritura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PA
      //--- escribimos el tiempo y los valores de precios de cierre en el archivo
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteFloat(file_handle,(float)close_buff[i]);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos grabados, el archivo %s cerrado",InpFileName);
     }
   else
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

Véase también

Tipos reales (double, float), FileWriteDouble

© 2000-2025, MetaQuotes Ltd.

FileWriteInteger

Escribe el valor del parámetro del tipo int desde la posición actual del puntero de archivos en un archivo binario.

```
uint  FileWriteInteger(
   int  file_handle,         // manejador del archivo
   int  value,               // valor para escribir
   int  size=INT_VALUE       // tamaño en bytes
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

value

[in] Valor entero.

size=INT_VALUE

[in] Cantidad de bytes (hasta 4 inclusive) que hay que escribir. Están previstas las constantes siguientes: CHAR_VALUE=1, SHORT_VALUE=2 y INT_VALUE=4, de esta manera, la función puede escribir el valor entero del tipo char, uchar, short, ushort, int o uint.

Valor devuelto

En caso del éxito la función devuelve el número de bytes escritos. El puntero de archivos se mueve a esta misma cantidad de bytes.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                      Demo_FileWriteInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros para obtener datos desde el terminal
input string             InpSymbolName="EURUSD";           // par de divisas
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // período de tiempo
input datetime           InpDateStart=D'2013.01.01 00:00'; // fecha de inicio del copi
//--- parámetros para la escritura de datos en el archivo
input string             InpFileName="Trend.bin"; // nombre del archivo
input string             InpDirectoryName="Data"; // nombre de la carpeta
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- reseteamos el valor del error
   ResetLastError();
//--- copiamos el precio de cierre para cada barra
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Fallo al copiar los valores de los precios de cierre. Código del er
      return;
     }
//--- copiamos el tiempo para cada barra
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Fallo al copiar el valor del tiempo. Código del error = %d",GetLast
      return;
     }
//--- obtenemos el tamaño del búfer
   size=ArraySize(close_buff);
//--- abrimos el archivo para escribir los valores (si no existe, se crea automáticame
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("El archivo %s está abierto para la escritura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PA
      //---
      int   up_down=0; // bandera de la tendencia
      int   arr_size;  // tamaño del array arr
      uchar arr[];     // array del tipo uchar
      //--- escribimos los valores de tiempo en el archivo
      for(int i=0;i<size-1;i++)
        {
         //--- comparamos los precios de cierre para la barra actual y la siguiente
         if(close_buff[i]<=close_buff[i+1])
           {
            if(up_down!=1)
              {
```

© 2000-2025, MetaQuotes Ltd.

```
               //--- escribimos los valores de la fecha en el archivo utilizando FileW
               StringToCharArray(TimeToString(time_buff[i]),arr);
               arr_size=ArraySize(arr);
               //--- primero escribimos el número de símbolos del array
               FileWriteInteger(file_handle,arr_size,INT_VALUE);
               //--- escribimos los propios símbolos
               for(int j=0;j<arr_size;j++)
                  FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
               //--- cambiamos la bandera de la tendencia
               up_down=1;
              }
           }
         else
           {
            if(up_down!=-1)
              {
               //--- escribimos el valor de la fecha en el archivo utilizando FileWrit
               StringToCharArray(TimeToString(time_buff[i]),arr);
               arr_size=ArraySize(arr);
               //--- primero escribimos el número de símbolos del array
               FileWriteInteger(file_handle,arr_size,INT_VALUE);
               //--- escribimos los propios símbolos
               for(int j=0;j<arr_size;j++)
                  FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
               //--- cambiamos la bandera de la tendencia
               up_down=-1;
              }
           }
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos grabados, el archivo %s cerrado",InpFileName);
     }
   else
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
  }
```

Véase también

IntegerToString, StringToInteger, Tipos enteros

© 2000-2025, MetaQuotes Ltd.

FileWriteLong

Escribe el valor del parámetro del tipo long desde la posición actual del puntero de archivos en un archivo binario.

```
uint  FileWriteLong(
   int   file_handle,     // manejador del archivo
   long  value            // valor para escribir
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

value

[in] Valor del tipo long.

Valor devuelto

En caso del éxito la función devuelve el número de bytes escritos (en este caso sizeof(long)=8). El puntero de archivos se mueve a esta misma cantidad de bytes.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros para obtener datos desde el terminal
input string          InpSymbolName="EURUSD";           // par de divisas
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // período de tiempo
input datetime        InpDateStart=D'2013.01.01 00:00'; // fecha de inicio del copiado
//--- parámetros para la escritura de datos en el archivo
input string          InpFileName="Volume.bin"; // nombre del archivo
input string          InpDirectoryName="Data";  // nombre de la carpeta
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   long     volume_buff[];
   datetime time_buff[];
   int      size;
//--- reseteamos el valor del error
   ResetLastError();
//--- copiamos los volúmenes de ticks para cada barra
   if(CopyTickVolume(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,volume_buf
     {
      PrintFormat("Fallo al copiar los valores del volumen de ticks. Código del error
      return;
     }
//--- copiamos el tiempo para cada barra
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Fallo al copiar el valor del tiempo. Código del error = %d",GetLast
      return;
     }
//--- obtenemos el tamaño del búfer
   size=ArraySize(volume_buff);
//--- abrimos el archivo para escribir los valores del indicador (si no existe, se cre
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("El archivo %s está abierto para la escritura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PAT
      //--- primero escribimos el tamaño de la muestra de datos
      FileWriteLong(file_handle,(long)size);
      //--- escribimos el tiempo y los valores del volumen en el archivo
      for(int i=0;i<size;i++)
        {
         FileWriteLong(file_handle,(long)time_buff[i]);
         FileWriteLong(file_handle,volume_buff[i]);
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos grabados, el archivo %s cerrado",InpFileName);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
}
```

Véase también

Tipos enteros, FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileWriteString

Esta función escribe el valor del parámetro del tipo string en un archivo del tipo BIN, CSV o TXT desde la posición actual del puntero de archivos. En el caso de la escritura en los archivos del tipo CSV o TXT: si en la cadena se encuentra el símbolo '\n' (LF) sin el símbolo '\r' (CR) que le precede, entonces antes del símbolo '\n' se añade el correspondiente símbolo '\r'.

```
uint  FileWriteString(
   int           file_handle,     // manejador del archivo
   const string  text_string,     // cadena para escribir
   int           length=-1        // número de símbolos
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

text_string

[in] Cadena.

length=-1

[in] Número de símbolos que hay que escribir. Este parámetro es necesario para escribir una cadena en el archivo del tipo BIN. Si el tamaño no está especificado, se escribe la cadena entera sin el 0 final. Si el tamaño especificado es menos que longitud de la cadena, se escribe una parte de la cadena sin 0 final. Si el tamaño especificado es más que longitud de la cadena, entonces la cadena se complementa con la cantidad de ceros correspondiente. Este parámetro se ignora para los archivos CSV y TXT y la cadena se escribe en su totalidad.

Valor devuelto

En caso del éxito la función devuelve el número de bytes escritos. El puntero de archivos se mueve a esta misma cantidad de bytes.

Nota

Hay que tener en cuenta que cuando escribimos en un archivo abierto con la bandera FILE_UNICODE (o sin bandera FILE_ANSI), el número de bytes escritos será doble del número de caracteres de cadena escritos. Cuando escribimos en un archivo abierto con la bandera FILE_ANSI, el número de bytes escritos va a coincidir con el número de símbolos de cadena que han sido escritos.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteString.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros para obtener datos desde el terminal
input string             InpSymbolName="EURUSD";           // par de divisas
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // período de tiempo
input int                InpMAPeriod=14;                   // período de la media móvi
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // tipo del precio
input datetime           InpDateStart=D'2013.01.01 00:00'; // fecha de inicio del copi
//--- parámetros para la escritura de datos en el archivo
input string             InpFileName="RSI.csv";    // nombre del archivo
input string             InpDirectoryName="Data"; // nombre de la carpeta
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // fecha del fin de copiado de datos
   double   rsi_buff[];  // array de los valores del indicador
   datetime date_buff[]; // array de las fechas del indicador
   int      rsi_size=0;  // tamaño de arrays del indicador
//--- tiempo de finalización - actual
   date_finish=TimeCurrent();
//--- obtenemos el manejador del indicador RSI
   ResetLastError();
   int rsi_handle=iRSI(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- fallo al obtener el manejador del indicador
      PrintFormat("Error al obtener el manejador del indicador. Código del error = %d"
      return;
     }
//--- seguimos en el ciclo hasta que el indicador no termine de calcular todos sus val
   while(BarsCalculated(rsi_handle)==-1)
      Sleep(10); // retardo para que al indicador le de tiempo a calcular sus valores
//--- copiamos los valores del indicador de un período determinado
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,date_finish,rsi_buff)==-1)
     {
      PrintFormat("Fallo al copiar los valores del indicador. Código del error = %d",G
      return;
     }
//--- copiamos el tiempo correspondiente para los valores del indicador
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("Fallo al copiar los valores del tiempo. Código del error = %d",GetL
      return;
     }
//--- liberamos la memoria que ocupa el indicador
   IndicatorRelease(rsi_handle);
//--- obtenemos el tamaño del búfer
   rsi_size=ArraySize(rsi_buff);
//--- abrimos el archivo para escribir los valores del indicador (si no existe, se cre
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("El archivo %s está abierto para la escritura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PA
      //--- preparamos variables auxiliares
      string str="";
      bool   is_formed=false;
      //--- escribimos las fechas de formación de las zonas de sobrecompra y sobrevent
      for(int i=0;i<rsi_size;i++)
        {
         //--- prueba de los valores del indicador
         if(rsi_buff[i]>=70 || rsi_buff[i]<=30)
           {
            //--- si es el primer valor en esta área
            if(!is_formed)
              {
               //--- añadimos el valor y la fecha
               str=(string)rsi_buff[i]+"\t"+(string)date_buff[i];
               is_formed=true;
              }
            else
               str+="\t"+(string)rsi_buff[i]+"\t"+(string)date_buff[i];
            //--- paso a la siguiente iteración del ciclo
            continue;
           }
         //--- prueba de la bandera
         if(is_formed)
           {
            //--- la cadena está formada, la escribimos en el archivo
            FileWriteString(file_handle,str+"\r\n");
            is_formed=false;
           }
        }
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos grabados, el archivo %s cerrado",InpFileName);
     }
   else
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
  }
```

Véase también

Tip string, StringFormat

© 2000-2025, MetaQuotes Ltd.

FileWriteStruct

Escribe el contenido de una estructura pasada como un parámetro en un archivo binario desde la posición actual del puntero de archivos.

```
uint  FileWriteStruct(
   int           file_handle,       // manejador del archivo
   const void&   struct_object,     // enlace a objeto
   int           size=-1            // tamaño para escribir en bytes
   );
```

Parámetros

file_handle

[in] Descriptor de archivo devuelto por la función FileOpen().

struct_object

[in] Referencia al objeto de dicha estructura. La estructura no debe contener líneas de caracteres, matrices dinámicas, funciones virtuales u objetos de clases, así como punteros a objetos y funciones.

size=-1

[in] Número de símbolos que hay que escribir. Si el tamaño no está especificado o está indicada la cantidad de bytes que supera el tamaño de la estructura, se escribe la estructura entera.

Valor devuelto

En caso del éxito la función devuelve el número de bytes escritos. El puntero de archivos se mueve a esta misma cantidad de bytes.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWiteStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- parámetros para obtener datos desde el terminal
input string          InpSymbolName="EURUSD";           // par de divisas
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // período de tiempo
input datetime        InpDateStart=D'2013.01.01 00:00'; // fecha de inicio del copiado
//--- parámetros para la escritura de datos en el archivo
input string          InpFileName="EURUSD.txt";         // nombre del archivo
input string          InpDirectoryName="Data";          // nombre de la carpeta
//+------------------------------------------------------------------+
//| Estructura para almacenar los datos de la vela                   |
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // precio de apertura
   double            close; // precio de cierre
   double            high;  // precio máximo
   double            low;   // precio mínimo
   datetime          date;  // fecha
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime     date_finish=TimeCurrent();
   int          size;
   datetime     time_buff[];
   double       open_buff[];
   double       close_buff[];
   double       high_buff[];
   double       low_buff[];
   candlesticks cand_buff[];
//--- reseteamos el valor del error
   ResetLastError();
//--- obtenemos el tiempo de aparición de barras desde el rango
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Fallo al copiar el valor de tiempo. Código del error = %d",GetLastE
      return;
     }
//--- obtenemos los precios máximos de las barras desde el rango
   if(CopyHigh(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,high_buff)==-1)
     {
      PrintFormat("Fallo al copiar los valores de precios máximos. Código del error =
      return;
     }
//--- obtenemos los precios mínimos de las barras desde el rango
   if(CopyLow(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,low_buff)==-1)
     {
      PrintFormat("Fallo al copiar los valores de precios mínimos. Código del error =
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- obtenemos los precios de apertura de barras desde al rango
   if(CopyOpen(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,open_buff)==-1)
     {
      PrintFormat("Fallo al copiar los valores de precios de apertura. Código del erro
      return;
     }
//--- obtenemos los precios de cierre de barras desde el rango
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Fallo al copiar los valores de precios de cierre. Código del error
      return;
     }
//--- definimos la dimensión de arrays
   size=ArraySize(time_buff);
//--- guardamos todos los datos en el array de la estructura
   ArrayResize(cand_buff,size);
   for(int i=0;i<size;i++)
     {
      cand_buff[i].open=open_buff[i];
      cand_buff[i].close=close_buff[i];
      cand_buff[i].high=high_buff[i];
      cand_buff[i].low=low_buff[i];
      cand_buff[i].date=time_buff[i];
     }

//--- abrimos el archivo para la escritura del array de la estructura en el archivo (s
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIl
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("El archivo %s está abierto para la escritura",InpFileName);
      PrintFormat("Ruta del archivo: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDA
      //--- preparamos el contador del número de bytes
      uint counter=0;
      //--- escribimos los valores del array usando el ciclo
      for(int i=0;i<size;i++)
         counter+=FileWriteStruct(file_handle,cand_buff[i]);
      PrintFormat("En el archivo %s han sido escritos %d bytes de información",InpFile
      PrintFormat("Total de bytes: %d * %d * %d = %d, %s",size,5,8,size*5*8,size*5*8==
      //--- cerramos el archivo
      FileClose(file_handle);
      PrintFormat("Datos escritos, archivo %s cerrado",InpFileName);
     }
   else
      PrintFormat("Fallo al abrir el archivo %s, Código del error = %d",InpFileName,Ge
  }
```

Véase también

Estructuras y clases

© 2000-2025, MetaQuotes Ltd.

FileLoad

Lee todo el contenido de un archivo binario especificado en una matriz transmitida de tipos numéricos o estructuras sencillas. La función permite leer rápidamente los datos de un tipo conocido en la matriz apropiada.

```
long  FileLoad(
   const string  file_name,          // nombre del archivo
   void&         buffer[],           // matriz de tipos numéricos o estructuras sencil
   int           common_flag=0       // bandera del archivo, por defecto el archivo se
   );
```

Parámetros

file_name

[in] Nombre del archivo desde el que se leerán los datos.

buffer

[out] Matriz de tipos numéricos o estructuras sencillas.

common_flag=0

[in] bandera de archivo que indica el modo de trabajo. Si no se indica el parámetro, entonces el archivo se buscará en la subcarpeta MQL5\Files (o <catálogo_del_agente_de_simulación>\MQL5\Files en caso de simulación).

Valor devuelto

Número de elementos leídos o -1 en caso de fallo.

Observación

La función FileLoad() lee del archivo un número de bytes múltiplo del tamaño del elemento de la matriz. Por ejemplo, supongamos que el tamaño del archivo es de 10 bytes, y la lectura se realiza en una matriz de tipo double (sizeof(double)=8). En este caso, solo se leerán 8 bytes, los 2 restantes al final del archivo sencillamente se descartarán, y la propia función FileLoad() retornará 1 (1 elemento leído).

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileLoad.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- input parameters
input int      bars_to_save=10; // número de barras
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_rates.bin";
   MqlRates rates[];
//---
   int copied=CopyRates(_Symbol,_Period,0,bars_to_save,rates);
   if(copied!=-1)
     {
      PrintFormat(" CopyRates(%s) copied %d bars",_Symbol,copied);
      //---  escribimos las cotizaciones en el archivo
      if(!FileSave(filename,rates,FILE_COMMON))
         PrintFormat("FileSave() failed, error=%d",GetLastError());
     }
   else
      PrintFormat("Failed CopyRates(%s), error=",_Symbol,GetLastError());
//--- ahora leemos estas cotizaciones de vuelta desde el archivo
   ArrayFree(rates);
   long count=FileLoad(filename,rates,FILE_COMMON);
   if(count!=-1)
     {
      Print("Time\tOpen\tHigh\tLow\tClose\tTick Voulme\tSpread\tReal Volume");
      for(int i=0;i<count;i++)
        {
         PrintFormat("%s\t%G\t%G\t%G\t%G\t%I64u\t%d\t%I64u",
                     TimeToString(rates[i].time,TIME_DATE|TIME_SECONDS),
                     rates[i].open,rates[i].high,rates[i].low,rates[i].close,
                     rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
  }
```

Véase también

Estructuras y clases, FileReadArray, FileReadStruct, FileSave

© 2000-2025, MetaQuotes Ltd.

FileSave

Guarda en un archivo binario todos los elementos de la matriz transmitida como parámetro. La función permite anotar rápidamente en una línea las matrices de los tipos numéricos o de las estructuras sencillas.

```
bool  FileSave(
   const string  file_name,          // nombre del archivo
   void&         buffer[],           // matriz de tipos numéricos o estructuras sencill
   int           common_flag=0       // bandera del archivo, por defecto los archivos s
   );
```

Parámetros

file_name

[in] Nombre del archivo en el que se escribirá la matriz de datos.

buffer

[in] Matriz de tipos numéricos o estructuras sencillas.

common_flag=0

[in] bandera de archivo que indica el modo de trabajo. Si no se indica el parámetro, entonces el archivo se escribirá en la subcarpeta MQL5\Files (o <catálogo_del_agente_de_simulación>\MQL5\Files en caso de simulación).

Valor devuelto

En caso de fallo, la función retorna false.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSave.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- input parameters
input int      ticks_to_save=1000; // número de ticks
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_ticks.bin";
   MqlTick ticks[];
//---
   int copied=CopyTicks(_Symbol,ticks,COPY_TICKS_ALL,0,ticks_to_save);
   if(copied!=-1)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat(" CopyTicks(%s) copied %d ticks",_Symbol,copied);
       //--- si la historia de ticks está sincronizada, entonces el código de error es
       if(!GetLastError()==0)
          PrintFormat("%s: Ticks are not synchronized, error=%d",_Symbol,copied,_LastEr
       //---  escribimos los ticks en el archivo
       if(!FileSave(filename,ticks,FILE_COMMON))
          PrintFormat("FileSave() failed, error=%d",GetLastError());
      }
    else
       PrintFormat("Failed CopyTicks(%s), Error=",_Symbol,GetLastError());
//--- ahora leemos estos ticks de vuelta desde el archivo
    ArrayFree(ticks);
    long count=FileLoad(filename,ticks,FILE_COMMON);
    if(count!=-1)
      {
       Print("Time\tBid\tAsk\tLast\tVolume\tms\tflags");
       for(int i=0;i<count;i++)
         {
          PrintFormat("%s.%03I64u:\t%G\t%G\t%G\t%I64u\t0x%04x",
          TimeToString(ticks[i].time,TIME_DATE|TIME_SECONDS),ticks[i].time_msc%1000,
          ticks[i].bid,ticks[i].ask,ticks[i].last,ticks[i].volume,ticks[i].flags);
         }
      }
  }
```

Véase también

Estructuras y clases, FileWriteArray, FileWriteStruct, FileLoad, FileWrite

© 2000-2025, MetaQuotes Ltd.

FolderCreate

Crea una carpeta en el directorio Files (dependiendo del valor de common_flag)

```
bool  FolderCreate(
   string  folder_name,       // cadena con el nombre de la carpeta a crear
   int     common_flag=0      // zona de alcance
   );
```

Parámetros

folder_name

[in] Nombre de la carpeta que hay que crear. Contiene la ruta relativa a la carpeta.

common_flag=0

[in] Bandera que determina la ubicación de la carpeta. Si es common_flag=FILE_COMMON, la carpeta se encuentra en la carpeta compartida de todos los terminales de cliente \Terminal\Common\Files. De lo contrario, la carpeta se encuentra en la carpeta local (MQL5\files o MQL5\tester\files en caso de prueba).

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Ejemplo:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- descripción
#property description "El script muestra un ejemplo de uso de FolderCreate()."
#property description "El parámetro externo determina la carpeta para la creación de c
#property description "Después de ejecutar el script, se creará una estructura de carp

//--- mostramos la ventana de parámetros de entrada al iniciar el script
#property script_show_inputs
//--- el parámetro de entrada determina la carpeta en la que funciona el script
input bool     common_folder=false; // carpeta general de todos los terminales
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- carpeta que crearemos en MQL5\Files
   string root_folder="Folder_A";
   if(CreateFolder(root_folder,common_folder))
     {
      //--- creamos en ella la carpeta hija Child_Folder_B1
      string folder_B1="Child_Folder_B1";
      string path=root_folder+"\\"+folder_B1;          // creamos el nombre de la carp
      if(CreateFolder(path,common_folder))
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- en esta carpeta creamos otras 3 carpetas hijas
          string folder_C11="Child_Folder_C11";
          string child_path=root_folder+"\\"+folder_C11;// creamos el nombre de la carp
          CreateFolder(child_path,common_folder);
          //--- segunda carpeta hija
          string folder_C12="Child_Folder_C12";
          child_path=root_folder+"\\"+folder_C12;
          CreateFolder(child_path,common_folder);

          //--- tercera carpeta hija
          string folder_C13="Child_Folder_C13";
          child_path=root_folder+"\\"+folder_C13;
          CreateFolder(child_path,common_folder);
         }
      }
//---
  }
//+------------------------------------------------------------------+
//| Intenta crear una carpeta y muestra un mensaje sobre ello        |
//+------------------------------------------------------------------+
bool CreateFolder(string folder_path,bool common_flag)
  {
   int flag=common_flag?FILE_COMMON:0;
   string working_folder;
//--- aclaramos la ruta completa dependiendo del parámetro common_flag
   if(common_flag)
      working_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH)+"\\MQL5\\Files";
   else
      working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- mensaje de depuración
   PrintFormat("folder_path=%s",folder_path);
//---intentando crear una carpeta relativa a la ruta MQL5\Files
   if(FolderCreate(folder_path,flag))
     {
      //--- mostramos la ruta completa para la carpeta creada
      PrintFormat("Hemos creado la carpeta %s",working_folder+"\\"+folder_path);
      //--- reseteamos el código de error
      ResetLastError();
      //--- ejecutado con éxito
      return true;
     }
   else
      PrintFormat("No se ha logrado crear la carpeta %s. Código de error %d",working_f
//--- ejecutado sin éxito
   return false;
  }
```

Véase también

© 2000-2025, MetaQuotes Ltd.

FileOpen(), FolderClean(), FileCopy()

© 2000-2025, MetaQuotes Ltd.

FolderDelete

Elimina el directorio especificado. Si la carpeta no está vacía, no se puede eliminarla.

```
bool  FolderDelete(
   string  folder_name,       // cadena con el nombre de la carpeta a eliminar
   int     common_flag=0      // zona de alcance
   );
```

Parámetros

folder_name

[in] Nombre del directorio que hay que eliminar. Contiene la ruta entera hacia la carpeta.

common_flag=0

[in] Bandera que determina la ubicación de la carpeta. Si es common_flag=FILE_COMMON, la carpeta se encuentra en la carpeta compartida de todos los terminales de cliente \Terminal\Common\Files. De lo contrario, la carpeta se encuentra en la carpeta local (MQL5\files o MQL5\tester\files en caso de prueba).

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

Si el directorio contiene al menos un archivo y/o un subdirectorio, es imposible eliminar este directorio; previamente hay que desocuparlo. La liberación total de una carpeta de todos los archivos y subcarpetas anidadas se realiza mediante la función FolderClean().

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              Demo_FolderDelete.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://team.metaquotes.ru/email/view/599588"
#property version   "1.00"
//--- descripción
#property description "El script muestra el ejemplo del uso de FolderDelete()."
#property description "Primero se crean dos carpetas: una está vacía, la otra contiene
#property description "Cuando se intenta eliminar la carpeta no vacía, se devuelve el

//--- mostraremos la ventana de los parámetros de entrada durante el inicio del script
#property script_show_inputs
//--- parámetros de entrada
input string   firstFolder="empty";     // carpeta vacía
input string   secondFolder="nonempty";// carpeta que va a contener un archivo
string filename="delete_me.txt";       // nombre del archivo que vamos a crear en la c
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- escribiremos el manejador del archivo aquí
   int handle;
//--- buscaremos la carpeta en la que trabajamos
   string working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- mensaje de depuración
   PrintFormat("working_folder=%s",working_folder);
//--- intento de crear una carpeta vacía respecto a la ruta MQL5\Files
   if(FolderCreate(firstFolder,0)) // 0 significa que estamos trabajando en la carpeta
     {
      //--- mostraremos la ruta completa de la carpeta creada
      PrintFormat("Hemos creado la carpeta %s",working_folder+"\\"+firstFolder);
      //--- anularemos el código del error
      ResetLastError();
     }
   else
      PrintFormat("Fallo al crear la carpeta %s. Código del error %d",working_folder+"

//--- ahora crearemos una carpeta no vacía usando la función FileOpen()
   string filepath=secondFolder+"\\"+filename;  // formaremos la ruta para el archivo
   handle=FileOpen(filepath,FILE_WRITE|FILE_TXT); // la bandera FILE_WRITE es obligato
   if(handle!=INVALID_HANDLE)
      PrintFormat("El archivo %s ha sido abierto para la lectura",working_folder+"\\"+
   else
      PrintFormat("Fallo al crear el archivo %s en la carpeta %s. Código del error=",f

   Comment(StringFormat("Preparándose para eliminar las carpetas %s y %s", firstFolder
//--- Una pequeña pausa de 5 segundos para que nos de tiempo a leer el mensaje en el g
   Sleep(5000); // ¡No se puede usar Sleep() en los indicadores!

//--- mostramos la ventana de diálogo y pedimos al usuario
   int choice=MessageBox(StringFormat("¿Desea eliminar las carpetas %s y %s?", firstFo
                         "Eliminando carpetas",
                         MB_YESNO|MB_ICONQUESTION); //  habrá dos botones - "Yes" y "N

//--- ejecutaremos la acción en función de la opción escogida
   if(choice==IDYES)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- quitamos el comentario del gráfico
      Comment("");
      //--- mostraremos el mensaje en el diario "Asesores Expertos"
      PrintFormat("Intentamos eliminar las carpetas %s y %s",firstFolder, secondFolder
      ResetLastError();
      //--- eliminamos la carpeta vacía
      if(FolderDelete(firstFolder))
         //--- debe aparecer el siguiente mensaje porque la carpeta está vacía
         PrintFormat("La carpeta %s ha sido eliminada con éxito",firstFolder);
      else
         PrintFormat("Fallo al eliminar la carpeta %s. Código del error=%d", firstFolc

      ResetLastError();
      //--- eliminamos la carpeta que contiene el archivo
      if(FolderDelete(secondFolder))
         PrintFormat("La carpeta %s ha sido eliminada con éxito", secondFolder);
      else
         //--- debe aparecer este mensaje porque hay un archivo dentro de la carpeta
         PrintFormat("Fallo al eliminar la carpeta %s. Código del error=%d", secondFol
     }
   else
      Print("Eliminación cancelada");
//---
  }
```

Véase también

FileOpen(), FolderClean(), FileMove()

© 2000-2025, MetaQuotes Ltd.

FolderClean

Elimina todos los archivos en una carpeta especificada.

```
bool  FolderClean(
   string  folder_name,       // cadena con el nombre de subcarpeta
   int     common_flag=0      // zona de alcance
   );
```

Parámetros

folder_name

[in] Nombre del directorio donde hay que eliminar todos los archivos. Contiene la ruta entera hacia la carpeta.

common_flag=0

[in] Bandera que determina la ubicación de la carpeta. Si es common_flag=FILE_COMMON, la carpeta se encuentra en la carpeta compartida de todos los terminales de cliente \Terminal\Common\Files. De lo contrario, la carpeta se encuentra en la carpeta local (MQL5\files o MQL5\tester\files en caso de prueba).

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

Cuidado con usar esta función porque todos los archivos y todos los subdirectorios anidados se eliminan sin que se pueda recuperarlos.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                Demo_FolderClean.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://team.metaquotes.ru/email/view/599588"
#property version   "1.00"
//--- descripción
#property description "Este script muestra un ejemplo de uso de FolderClean()."
#property description "Primero se crean los archivos en la carpeta especificada utiliz
#property description "Luego, antes de eliminar los archivos, se muestra el aviso util

//--- mostraremos la ventana de los parámetros de entrada durante el inicio del script
#property script_show_inputs
//--- parámetros de entrada
input string   foldername="demo_folder";  // crearemos la carpeta en MQL5/Files/
input int      files=5;                   // número de archivos que vamos a crear y el
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string name="testfile";
//--- primero abrimos o creamos archivos en la carpeta de datos de nuestro terminal
   for(int N=0;N<files;N++)
     {
      //--- nombre del archivo como 'demo_folder\testfileN.txt'
      string filemane=StringFormat("%s\\%s%d.txt",foldername,name,N);
      //--- abrimos el archivo con la bandera para escribir, en este caso la carpeta '
      int handle=FileOpen(filemane,FILE_WRITE);
      //--- veremos si la función FileOpen() ha trabajado con éxito
      if(handle==INVALID_HANDLE)
        {
         PrintFormat("Fallo al crear el archivo %s. Código del error ",filemane,GetLas
         ResetLastError();
        }
      else
        {
         PrintFormat("El archivo %s ha sido abierto con éxito",filemane);
         //--- ya no necesitamos el archivo abierto, hay que cerrarlo sí o sí
         FileClose(handle);
        }
     }

//--- comprobaremos el número de archivos en la carpeta
   int k=FilesInFolder(foldername+"\\*.*",0);
   PrintFormat("En total, la carpeta %s contiene %d archivos",foldername,k);

//--- mostramos la ventana de diálogo y preguntamos al usuario
   int choice=MessageBox(StringFormat("¿Desea eliminar de la carpeta %s %d archivos, c
                         "Eliminando archivos de la carpeta",
                         MB_YESNO|MB_ICONQUESTION); //  habrá dos botones - "Yes" y "N
   ResetLastError();

//--- ejecutaremos acciones en función de la opción seleccionada
   if(choice==IDYES)
     {
      //--- empezamos a eliminar
      PrintFormat("Intento de eliminar todos los archivos de la carpeta %s",foldername
      if(FolderClean(foldername,0))
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("Archivos eliminados con éxito, en la carpeta %s quedan %d archiv
                     foldername,
                     FilesInFolder(foldername+"\\*.*",0));
      else
         PrintFormat("Fallo al eliminar los archivos de la carpeta %s. Código del erro
     }
   else
      PrintFormat("Eliminación cancelada");
//---
  }
//+------------------------------------------------------------------+
//| devuelve el número de archivos en la carpeta especificada        |
//+------------------------------------------------------------------+
int FilesInFolder(string path,int flag)
  {
   int count=0;
   long handle;
   string filename;
//---
   handle=FileFindFirst(path,filename,flag);
//--- si ha sido encontrado por lo menos un archivo, seguimos buscando el resto
   if(handle!=INVALID_HANDLE)
     {
      //--- mostraremos el nombre del archivo
      PrintFormat("ha sido encntrado el archivo %s",filename);
      //--- aumentaremos el contador de archivos/carpetas encontrados
      count++;
      //--- iniciamos la búsqueda en todos los archivos/carpetas
      while(FileFindNext(handle,filename))
        {
         PrintFormat("ha sido encontrado el archivo %s",filename);
         count++;
        }
      //--- es obligatorio cerrar el manejador del buscador al finalizar
      FileFindClose(handle);
     }
   else // fallo al obtener el manejador
     {
      PrintFormat("Fallo al realizar la búsqueda de archivos en la carpeta %s",path);
     }
//--- devolveremos el resultado
   return count;
  }
```

Véase también

FileFindFirst, FileFindNext, FileFindClose

© 2000-2025, MetaQuotes Ltd.

Indicadores personalizados

Grupo de funciones que se usan para crear los indicadores personalizados. No se puede usarlas para crear los Asesores Expertos y los scripts.

Función	Acción
SetIndexBuffer	Enlaza el búfer de indicadores especificado con el array dinámico unidimensional del tipo double
IndicatorSetDouble	Establece el valor de una propiedad del indicador que tiene el tipo double
IndicatorSetInteger	Establece el valor de una propiedad del indicador que tiene el tipo int
IndicatorSetString	Establece el valor de una propiedad del indicador que tiene el tipo string
PlotIndexSetDouble	Establece el valor de propiedad de línea del indicador que tiene el tipo double
PlotIndexSetInteger	Establece el valor de propiedad de línea del indicador que tiene el tipo int
PlotIndexSetString	Establece el valor de propiedad de línea del indicador que tiene el tipo string
PlotIndexGetInteger	Devuelve el valor de propiedad de línea del indicador que tiene el tipo entero

Las propiedades de los indicadores se puede establecer tanto utilizando las directivas del compilador, como a través de las funciones. Para el mejor entendimiento del asunto, se recomienda estudiar los estilos de los indicadores en los ejemplos.

Todos los cálculos necesarios de los indicadores personalizados hay que colocar en la función predeterminada OnCalculate(). Si se usa la forma breve de la llamada a la función OnCalculate() del tipo

```
int OnCalculate (const int rates_total, const int prev_calculated, const int begin, co
```

entonces la variable *rates_total* contiene el valor de la cantidad total de elementos del array price[], que ha sido pasado como parámetro input para calcular valores del indicador.

El parámetro *prev_calculated* es el resultado de ejecución de la función OnCalculate() en la llamada anterior, permite organizar un algoritmo económico para calcular valores del indicador. Por ejemplo, si el valor actual rates_total=1000, y prev_calculated=999, entonces es posible que sea suficiente hacer los cálculos sólo para un valor de cada búfer de indicadores.

Si la información sobre el tamaño del array de entrada price no estuviera disponible, sería necesario hacer los cálculos para 1000 valores de cada búfer de indicador. Con la primera llamada a la función OnCalculate() el valor prev_calculated=0. Si de alguna manera el array price[] ha sido cambiado, en este caso prev_calculated también es igual a 0.

El parámetro *begin* dice el número de valores iniciales del array price que no contienen los datos para el cálculo. Por ejemplo, si los valores del indicador Accelerator Oscillator (para el que los primeros 37

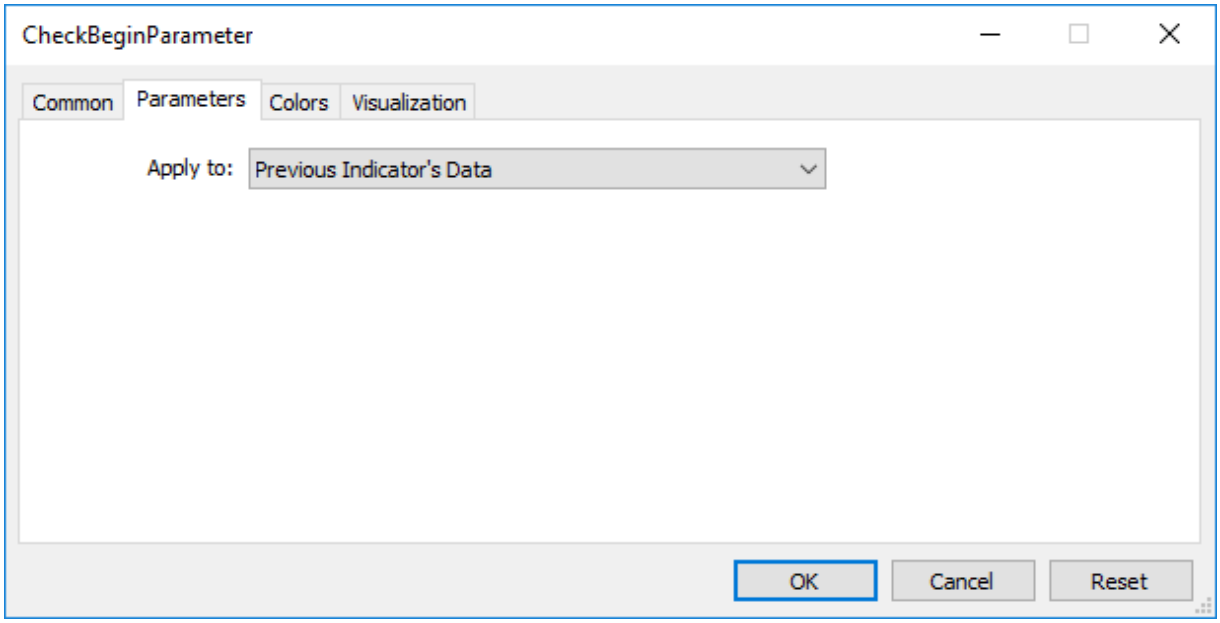
© 2000-2025, MetaQuotes Ltd.

valores no se calculan) han sido usados como un parámetro de entrada, entonces begin=37. Como ejemplo vamos a ver un indicador simple:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])

  {
//---
   Print("begin = ",begin,"  prev_calculated = ",prev_calculated,"  rates_total = ",ra
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Vamos a arrastrarlo desde la ventana "Navegador" a la ventana del indicador Accelerator Oscillator e indiquemos que los cálculos serán realizados a base de los valores del indicador anterior:

© 2000-2025, MetaQuotes Ltd.



Como resultado, con la primera llamada a la función OnCalculate() el valor prev_calculated va a ser igual a cero, y con las siguientes llamadas será igual al valor rates_total (hasta que el número de barras en el gráfico de precios no se aumente).



El valor del parámetro begin será exactamente igual al número de barras iniciales para las que los valores del indicador Accelerator no se calculan conforme a la lógica de este indicador. Si nos fijamos en el código fuente del indicador personalizado Accelerator.mq5, veremos en la función OnInit() estas líneas:

© 2000-2025, MetaQuotes Ltd.

```
//--- sets first bar from what index will be drawn
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,37);
```

Precisamente usando la función PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, empty_first_values) comunicamos el número de primeros valores inexistentes en el array de indicador nulo del indicador personalizado, los que no tenemos que considerar para el cálculo (empty_first_values). De esta manera, disponemos de mecanismos para:

1. comunicar sobre el número de valores iniciales de un indicador los que no hay que usar para los cálculos en otro indicador personalizado;
2. obtener la información sobre el número de primeros valores iniciales los que hay que ignorar a la hora de llamar a otro indicador personalizado sin entrar en la lógica de sus cálculos.

© 2000-2025, MetaQuotes Ltd.

Estilos de indicadores en ejemplos

El terminal de cliente MetaTrader 5 cuenta con 38 indicadores técnicos incorporados que se puede utilizar en los programas MQL5 a través de las funciones correspondientes. Pero el principal mérito del lenguaje MQL5 consiste en la posibilidad de crear sus propios indicadores personalizados que luego pueden ser utilizados en los Asesores Expertos para obtener valores, o simplemente pueden ser aplicados a los gráficos de precios para realizar el análisis técnico.

Usted puede conseguir toda la variedad de indicadores a base de unos estilos de dibujo básicos, llamados construcciones gráficas. Bajo la construcción se entiende el modo de visualización de datos que el indicador calcula, guarda y ofrece si se solicitan. Hay siete tipos de construcciones básicas:

1. línea,
2. sección (segmento),
3. histograma,
4. flecha (símbolo),
5. área coloreada (canal con relleno),
6. barras,
7. velas japonesas.

Cada construcción requiere para su visualización de uno a cinco arrays del tipo double en los que se guardan los valores del indicador. Para un cómodo trabajo del indicador, estos arrays se asocian a los búfers indicadores. La cantidad de búfers en el indicador hace falta declarar de antemano, utilizando las directivas del compilador, por ejemplo:

```
#property indicator_buffers 3 // número de búfers
#property indicator_plots   2 // número de construcciones gráficas
```

El número de búfers en el indicador siempre es igual o superior al número de construcciones gráficas en el indicador.

Puesto que cada construcción gráfica base puede tener diferentes variaciones de colores o un carácter específico de visualización, el número real de construcciones en el lenguaje MQL5 es 18:

Construcción	Descripción	Búfers de valores	Búfers de colores
DRAW_NONE	No se muestra visualmente en el gráfico, pero se puede ver los valores del búfer correspondiente	1	-

© 2000-2025, MetaQuotes Ltd.

Construcción	Descripción	Búfers de valores	Búfers de colores
	en la "Ventana de datos"		
DRAW_LINE	Se traza una línea a base de los valores del búfer correspondiente (los valores vacíos son indeseables en el búfer)	1	-
DRAW_SECTION	Se traza como una línea segmentada entre los valores del búfer correspondiente (normalmente hay muchos valores vacíos)	1	-
DRAW_HISTOGRAM	Se traza como un histograma desde la línea cero	1	-

© 2000-2025, MetaQuotes Ltd.

Construcción	Descripción	Búfers de valores	Búfers de colores
	hasta los valores del búfer correspondiente (puede tener valores vacíos)		
DRAW_HISTOGRAM2	Se traza como un histograma basándose en dos búfers indicadores (puede tener valores vacíos)	2	-
DRAW_ARROW	Se traza como símbolos (puede tener valores vacíos)	1	-
DRAW_ZIGZAG	Parecido al estilo DRAW_SECTION, pero a diferencia de éste puede dibujar segmentos verticale	2	-

© 2000-2025, MetaQuotes Ltd.

Construcción	Descripción	Búfers de valores	Búfers de colores
	s en una barra		
DRAW_FILLING	Relleno de colores entre dos líneas. En la "Ventana de datos" se muestran 2 valores de los búfers correspondientes	2	-
DRAW_BARS	Se visualiza en el gráfico en forma de las barras. En la "Ventana de datos" se muestran 4 valores de los búfers correspondientes	4	-
DRAW_CANDLES	Se visualiza en forma de las velas	4	-

© 2000-2025, MetaQuotes Ltd.

Construcción	Descripción	Búfers de valores	Búfers de colores
	japonesas. En la "Ventana de datos" se muestran 4 valores de los búfers correspondientes		
DRAW_COLOR_LINE	Es una línea para la que se puede alternar colores en diferentes barras y cambiar su calor en el momento deseado	1	1
DRAW_COLOR_SECTION	Parecido al estilo DRAW_SECTION, pero el color para cada sección se puede definir de manera individual; también	1	1

© 2000-2025, MetaQuotes Ltd.

Construcción	Descripción	Búfers de valores	Búfers de colores
	se puede definir el color de forma dinámica		
DRAW_COLOR_HISTOGRAM	Parecido al estilo DRAW_HISTOGRAM, pero cada raya puede tener su propio color; también se puede definir el color de forma dinámica	1	1
DRAW_COLOR_HISTOGRAM2	Parecido a DRAW_HISTOGRAM2, ero cada raya puede tener su propio color; también se puede definir el color de forma	2	1

© 2000-2025, MetaQuotes Ltd.

Construcción	Descripción	Búfers de valores	Búfers de colores
	dinámica		
DRAW_COLOR_ARROW	Parecido al estilo DRAW_ARROW, pero cada símbolo puede tener su propio color. Se puede cambiar el color dinámicamente	1	1
DRAW_COLOR_ZIGZAG	El estilo DRAW_ZIGZAG con las posibilidades del coloreado individual de secciones y el cambio dinámico del color	2	1
DRAW_COLOR_BARS	Estilo DRAW_BARS con las posibilidades del coloreado individual de barras y el	4	1

© 2000-2025, MetaQuotes Ltd.

Construcción	Descripción	Búfers de valores	Búfers de colores
	cambio dinámico del color		
DRAW_COLOR_CANDLES	Estilo DRAW_CANDLES con las posibilidades del coloreado individual de velas y el cambio dinámico del color	4	1

La diferencia entre un búfer indicador y un array

En cada indicador hay que declarar a un nivel global uno o más arrays del tipo double que luego tendrá que ser utilizado como búfer indicador mediante la función SetIndexBuffer(). Para dibujar las construcciones gráficas del indicador, se utilizan sólo los valores desde el búfer indicador. No se puede utilizar ningún otro array para este propósito. Además de eso, los valores de los búfers se muestran en la "Ventana de datos".

Un búfer indicador tiene que ser dinámico y no requiere la especificación del tamaño – el tamaño del array que ha sido utilizado como búfer indicador se determina por el subsistema ejecutable del terminal de forma automática.

Tras la vinculación del array con el búfer indicador, la dirección de indexación se establece por defecto como en los arrays comunes, pero si hace falta, Usted puede aplicar la función ArraySetAsSeries() para cambiar el modo de acceso a los elementos del array. Por defecto, el búfer indicador se utiliza para almacenar los datos que están destinados para el proceso de dibujo (INDICATOR_DATA).

Si para el cálculo de los valores del indicador es necesario realizar algunas computaciones intermedias y almacenar un valor adicional para cada barra, entonces durante la vinculación este array puede ser declarado como el búfer de cálculo (INDICATOR_CALCULATIONS). Un array ordinario también puede ser utilizado para los valores intermedios, pero en este caso el programador debe encargarse personalmente de controlar el tamaño de este array.

Algunas construcciones permiten establecer un color de visualización para cada barra. Para almacenar la información sobre el color, se utilizan los búfers de colores (INDICATOR_COLOR_INDEX). El color

© 2000-2025, MetaQuotes Ltd.

está representado con el tipo de números enteros color, pero todos los búfers indicadores deben tener el tipo double. No se puede obtener los valores de colores y de los búfers auxiliares (INDICATOR_CALCULATIONS) mediante la función CopyBuffer().

El número de los búfers indicadores debe ser indicado por la directiva del compilador #property indicator_buffers número_de_búfers:

```
#property indicator_buffers 3  //  el indicador tiene 3 búfers
```

El número máximo permitido de los búfers en un indicador - 512.

Correspondencia de búfers indicadores y construcciones gráficas

Cada construcción gráfica se basa en uno o más búfers indicadores. De esta manera, para visualizar las velas japonesas simples, hacen falta cuatro valores: los precios Open, High, Low y Close. En consecuencia, para visualizar un indicador en forma de velas japonesas, es necesario declarar 4 búfers indicadores y 4 arrays del tipo double para ellos. Por ejemplo:

```
//--- el indicador tiene cuatro búfers indicadores
#property indicator_buffers 4
//--- el indicador tiene una construcción gráfica
#property indicator_plots   1
//--- la construcción gráfica con el número 1 se mostrará en forma de velas japonesas
#property indicator_type1   DRAW_CANDLES
//--- las velas japonesas serán dibujadas en color clrDodgerBlue
#property indicator_color1  clrDodgerBlue
//--- 4 arrays para los búfers indicadores
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

Las construcciones gráficas utilizan automáticamente los búfers indicadores de acuerdo con el número de construcción. Los números de construcción se empiezan desde 1, los números de búfers se empiezan desde 0. Si la primera construcción requiere 4 búfers indicadores, para el trazado serán utilizados los 4 primeros búfers indicadores. Estos cuatro búfers indicadores deben estar vinculados con los arrays correspondientes con la indexación correcta mediante la función SetIndexBuffer().

```
//--- Vinculación de arrays con búfers indicadores
   SetIndexBuffer(0,OBuffer,INDICATOR_DATA);  // el primer búfer corresponde al índice
   SetIndexBuffer(1,HBuffer,INDICATOR_DATA);  // el segundo búfer corresponde al índi
   SetIndexBuffer(2,LBuffer,INDICATOR_DATA);  // el tercer búfer corresponde al índice
   SetIndexBuffer(3,CBuffer,INDICATOR_DATA);  // el cuarto búfer corresponde al índice
```

Durante el dibujo de velas japonesas el indicador va a utilizar precisamente los cuatro primeros búfers, porque la construcción "velas japonesas" ha sido declarada bajo el número uno.

© 2000-2025, MetaQuotes Ltd.

Vamos a cambiar un poco nuestro ejemplo, vamos a añadir la construcción en forma de una línea simple - DRAW_LINE. Ahora, que la línea tenga el número 1, y las velas japonesas el número 2. El número de búfers y el número de construcciones se ha aumentado.

```
//--- el indicador tiene 5 búfers indicadores
#property indicator_buffers 5
//--- el indicador tiene 2 construcciones gráficas
#property indicator_plots   2
//--- la construcción gráfica con el número 1 se mostrará en forma de la línea
#property indicator_type1   DRAW_LINE
//--- la línea tendrá el color clrDodgerRed
#property indicator_color1  clrDodgerRed
//---la construcción gráfica con el número 2 se mostrará en forma de velas japonesas
#property indicator_type2   DRAW_CANDLES
//--- las velas japonesas serán dibujadas en color clrDodgerBlue
#property indicator_color2  clrDodgerBlue
//--- 5 arrays para los búfers indicadores
double LineBuffer[];
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

El orden de construcciones se ha cambiado, ahora primero va la línea, luego van las velas japonesas. Por esta razón, el orden de búfers va a ser el apropiado. Es decir, primero declararemos bajo el índice cero el búfer para la línea, y luego cuatro búfers para la visualización de velas japonesas.

```
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);  // el primer búfer corresponde al ínc
//--- vinculación de arrays con búfers indicadores para las velas japonesas
   SetIndexBuffer(1,OBuffer,INDICATOR_DATA);     // el segundo búfer corresponde al ín
   SetIndexBuffer(2,HBuffer,INDICATOR_DATA);     // el tercer búfer corresponde al ínc
   SetIndexBuffer(3,LBuffer,INDICATOR_DATA);     // el cuarto búfer corresponde al ínc
   SetIndexBuffer(4,CBuffer,INDICATOR_DATA);     // el quinto búfer corresponde al ínc
```

El número de búfers y construcciones gráficas se puede fijar sólo a través las directivas del compilador, es imposible el cambio dinámico de estas propiedades a través de las funciones.

Versiones de color de los estilos

Como se puede divisar de la tabla, los estilos se dividen en dos grupos. Al primer grupo pertenecen los estilos en cuyos nombres no figura la palabra **COLOR**, vamos a llamarlos estilos básicos:

- DRAW_LINE
- DRAW_SECTION
- DRAW_HISTOGRAM
- DRAW_HISTOGRAM2
- DRAW_ARROW
- DRAW_ZIGZAG

© 2000-2025, MetaQuotes Ltd.

- DRAW_FILLING
- DRAW_BARS
- DRAW_CANDLES

El segundo grupo de estilos contiene la palabra **COLOR**, vamos a llamarlos versiones de color:

- DRAW_COLOR_LINE
- DRAW_COLOR_SECTION
- DRAW_COLOR_HISTOGRAM
- DRAW_COLOR_HISTOGRAM2
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG
- DRAW_COLOR_BARS
- DRAW_COLOR_CANDLES

Todas las versiones de color de los estilos se diferencian de los temas básicos, lo que permite especificar un color para cada parte de la construcción. La parte mínima de la construcción es una barra, por eso se puede decir que las versiones de color permiten especificar un color para cada barra.

Para que se pueda especificar el color de la construcción en cada barra, en las versiones de color de los estilos ha sido agregado un búfer adicional especial para almacenar el índice del color. Estos índices indican el número del color en un array especial que contiene el conjunto de colores especificado de antemano. El tamaño del array de los colores - 64. Esto quiere decir que cada versión de color del estilo permite colorear una construcción gráfica con 64 diferentes colores.

El conjunto y el número de colores en este array especial se puede establecer con la directiva del compilador #property indicator_color, donde Usted puede especificar todos los colores necesarios separados con coma. Por ejemplo, hacemos la siguiente entrada en el indicador:

```
//--- estableceremos 8 colores para colorear las velas (se guardan en un array especia
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
```

Aquí está indicado que para la construcción gráfica número 1 hemos establecido 8 colores que serán colocados en el array especial. Luego en el programa no vamos a indicar el color para la construcción gráfica, sino vamos a utilizar su índice. Si queremos establecer para una barra el color rojo,para ello es necesario indicar en el búfer de colores el índice del color rojo desde el array. En la directiva el rojo va el primero, le corresponde el índice con el número 0.

```
  //--- fijaremos el color de la vela clrRed
  col_buffer[buffer_index]=0;
```

El conjunto de colores no es algo determinado de una vez y para siempre, se puede cambiarlo dinámicamente a través de la función PlotIndexSetInteger(). Ejemplo:

```
      //--- estableceremos el color para cada índice como la propiedad PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  número del estilo gráfico
                          PLOT_LINE_COLOR,      //  identificador de la propiedad
                          plot_color_ind,       //  índice del color donde escribiremo
                          color_array[i]);      //  nuevo color
```

© 2000-2025, MetaQuotes Ltd.

Propiedades del indicador y construcciones gráficas

Para las construcciones gráficas se puede establecer las propiedades tanto mediante las directivas del compilador, como a través de las funciones correspondientes. En el apartado Relación entre las propiedades de indicador y funciones correspondientes este asunto se describe con más detalles. El cambio dinámico de las propiedades del indicador a través de las funciones permite crear los indicadores personalizados más flexibles.

Inicio del proceso de dibujo de un indicador en el gráfico

En muchas ocasiones, según las condiciones del algoritmo, resulta imposible iniciar el cálculo de los valores del indicador inmediatamente desde la barra actual, es necesario proporcionar el número mínimo de las barras previas disponibles en el historial. Por ejemplo, muchos tipos del suavizado suponen el uso del array de precios para N barras anteriores, y a base de estos valores se calcula el valor del indicador para la barra actual.

En estas ocasiones, o no hay posibilidad de calcular los valores del indicador en las primeras N barras, o estos valores no están destinados para ser visualizados en el gráfico y conllevan un papel auxiliar para el cálculo de siguientes valores. Para renunciar la visualización del indicador en las primeras N barras del historial, se debe establecer el valor N para la propiedad PLOT_DRAW_BEGIN para la construcción gráfica correspondiente:

```
//--- vinculación de arrays con búfers indicadores para las velas japonesas
PlotIndexSetInteger(número_de_construcción_gráfica,PLOT_DRAW_BEGIN,N);
```

Aquí:

- número_de_construcción_gráfica - un valor de cero a indicator_plots-1 (la numeración de construcciones gráficas se empieza desde cero).
- N - número de las primeras barras en el historial en las cuales el indicador no debe mostrarse en el gráfico.

© 2000-2025, MetaQuotes Ltd.

DRAW_NONE

El estilo DRAW_NONE se utiliza cuando es necesario calcular los valores del búfer y mostrarlos en la "Ventana de datos", pero la misma representación en el gráfico no se precisa. Para ajustar la precisión de representación, utilice la expresión IndicatorSetInteger(INDICATOR_DIGITS,número_de_dígitos) en la función OnInit():

```
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- fijamos la precisión para el valor a mostrar en la Ventana de datos
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
```

El número de búfers requeridos para construir DRAW_NONE — 1.

Aquí tenemos un ejemplo del indicador que muestra en la "Ventana de datos" el número de la barra sobre la que está situado el ratón. La numeración corresponde a la serie temporal. Es decir, la barra actual no finalizada tiene el índice cero, mientras que la barra más antigua tiene el índice más grande.



Fíjense que a pesar de que la construcción gráfica №1 tenga especificado el color rojo de representación, el indicador no dibuja nada en el gráfico.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_NONE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Invisible
#property indicator_label1  "Bar Index"
#property indicator_type1   DRAW_NONE
#property indicator_style1  STYLE_SOLID
#property indicator_color1  clrRed
#property indicator_width1  1
//--- indicator buffers
double         InvisibleBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array con el búfer indicador
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- fijaremos la precisión con la que el valor será mostrado en la "Ventana de datos
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static datetime lastbar=0;
//--- si es el primer cálculo del indicador,
   if(prev_calculated==0)
     {
      //--- reenumeraremos las barras por primera vez
      CalcValues(rates_total,close);
      //--- recordaremos la hora de apertura de la barra actual en lastbar
      lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
    else
      {
       //--- si ha aparecido una barra nueva, la hora de su apertura no coincide con la
       if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
         {
          //--- volveremos a enumerar las barras de nuevo
          CalcValues(rates_total,close);
          //--- actualizaremos la hora de apertura de la barra actual en lastbar
          lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| enumeramos las barras como en una serie temporal                 |
//+------------------------------------------------------------------+
void CalcValues(int total,double const  &array[])
  {
//--- estableceremos para el búfer indicador la indexación como en la serie temporal
   ArraySetAsSeries(InvisibleBuffer,true);
//--- llenaremos para cada barra su número
   for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
  }
```

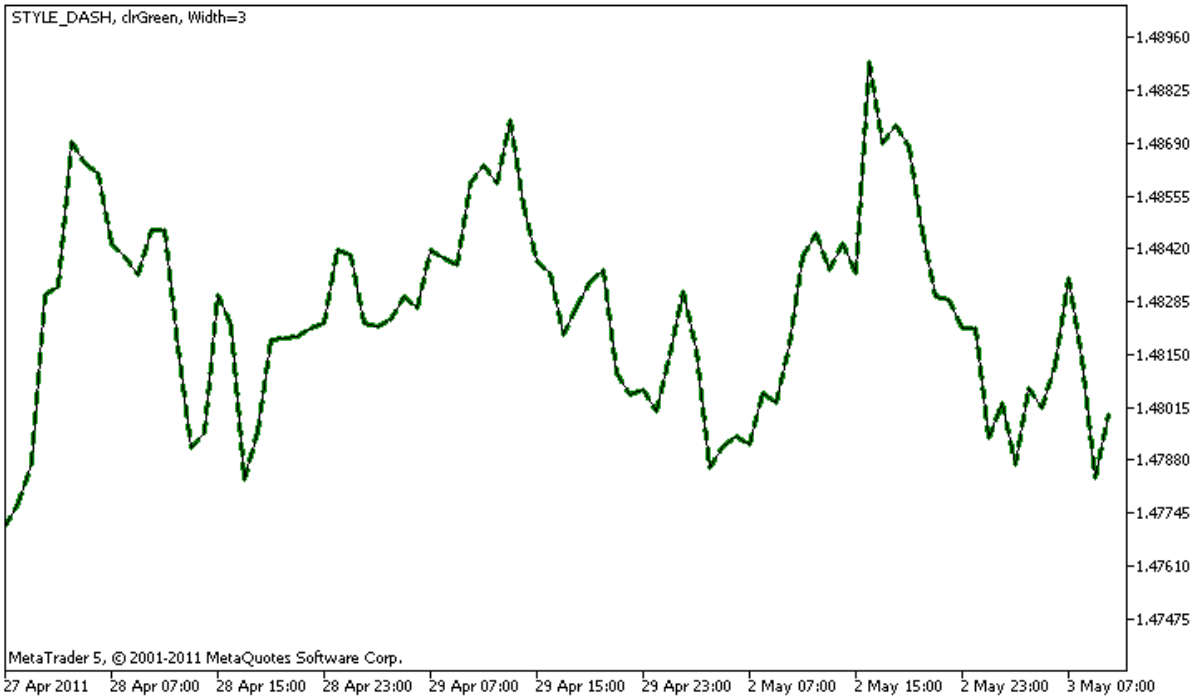
© 2000-2025, MetaQuotes Ltd.

DRAW_LINE

El estilo DRAW_LINE dibuja una línea de color especificado a base de los valores del búfer indicador. Usted puede establecer el grosor, color y el estilo de la línea tanto con las directivas del compilador, como de forma dinámica, utilizando la función PlotIndexSetInteger(). La opción del cambio dinámico de las propiedades de la construcción gráfica permite crear indicadores "vivos", es decir, los que cambian su apariencia en función de la situación actual.

El número de búfers requeridos para construir DRAW_LINE — 1.

Aquí tenemos un ejemplo de un indicador que traza una línea usando los precios de cierre de las barras Close. El color, grosor y el estilo de la línea se cambian de forma aleatoria cada N=5 tics.



Tenga presente que inicialmente para la construcción gráfica **plot1** con el estilo DRAW_LINE las propiedades se establecen mediante la directiva del compilador #property, y luego en la función OnCalculate() estas tres propiedades se establecen de forma aleatoria. El parámetro N está pasado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_LINE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_LINE"
#property description "Dibuja una línea de color especificado a base de los precios Cl
#property description "El color, grosor y el estilo de la línea se cambian de forma al
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "dentro de cada N tics"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- las propiedades de la línea están establecidas por medio de las directivas del c
#property indicator_label1  "Line"      // nombre de la construcción para la "Ventana
#property indicator_type1   DRAW_LINE   // tipo de construcción gráfica - una línea
#property indicator_color1  clrRed      // color de la línea
#property indicator_style1  STYLE_SOLID // estilo de la línea
#property indicator_width1  1           // grosor de la línea
//--- parámetro input
input int      N=5;          // número de tics para el cambio
//--- búfer indicador para la construcción
double         LineBuffer[];
//--- array para almacenar colores
color colors[]={clrRed,clrBlue,clrGreen};
//--- array para almacenar estilos de trazado de la línea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array con el búfer indicador
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- inicialización del generador de números pseudoaleatorios
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número crítico de tics,
```

© 2000-2025, MetaQuotes Ltd.

```
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- bloque para calcular los valores del indicador
   for(int i=0;i<rates_total;i++)
     {
      LineBuffer[i]=close[i];
     }

//--- volveremos el valor prev_calculated para la siguiente llamada de la función
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de la línea trazada en el indicador         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de la línea
   string comm="";
//--- bloque de cambio del color de la línea
//--- obtenemos un número aleatorio
   int number=MathRand();
//--- el divisor del número es igual al tamaño del array colors[]
   int size=ArraySize(colors);
//--- obtenemos el índice para seleccionar nuevo color como el remanente de la divisi
   int color_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- apuntaremos el color de la línea
   comm=comm+(string)colors[color_index];

//--- bloque de cambio del grosor de la línea
   number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5; // el grosor puede ser de 0 a 4
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+", Width="+IntegerToString(width);

//--- bloque de cambio del estilo de la línea
   number=MathRand();
//--- el divisor del número es igual al tamaño del array styles
   size=ArraySize(styles);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- obtenemos el índice para seleccionar nuevo estilo como el remanente de la divisi
   int style_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- apuntaremos el estilo de la línea
   comm=EnumToString(styles[style_index])+", "+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_SECTION

El estilo DRAW_SECTION dibuja los segmentos de color especificado a base de los valores del búfer indicador. El grosor, color y el estilo de la línea se puede establecer de la misma manera como para el estilo DRAW_LINE - con las directivas del compilador, o bien dinámicamente, utilizando la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite "vivificar" los indicadores, para que cambien su apariencia en función de la situación actual.

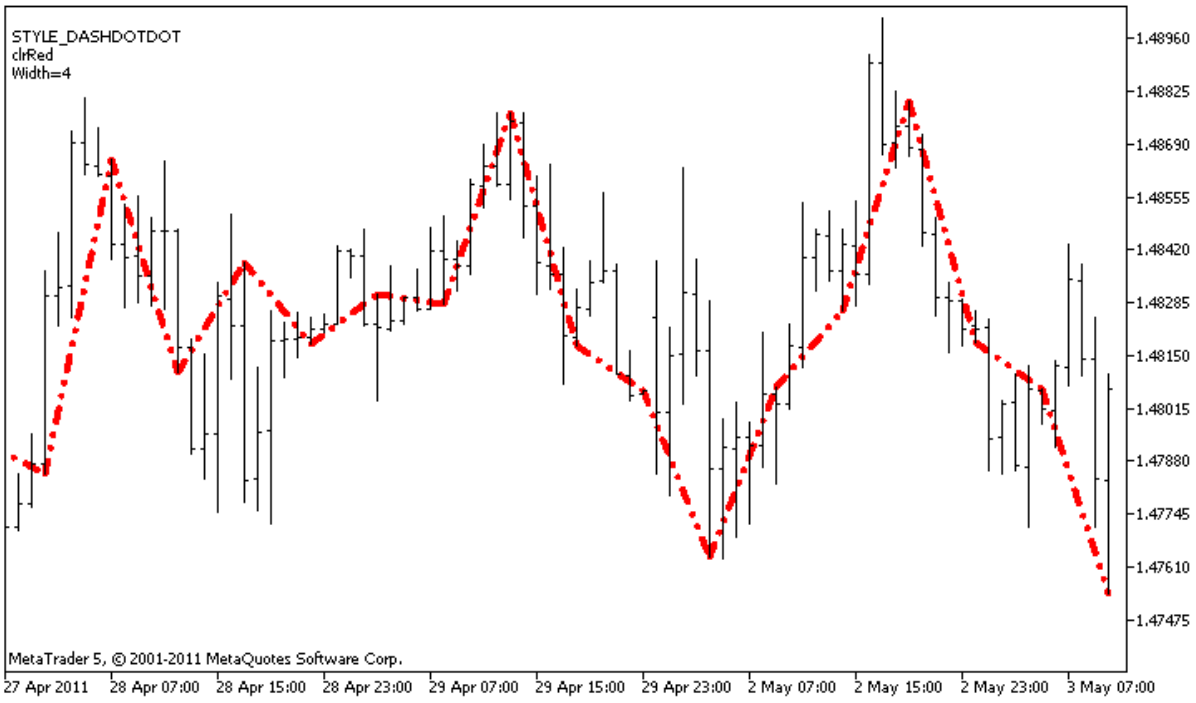
Los segmentos se trazan desde un valor no vacío hasta otro valor no vacío del búfer indicador, ignorando los valores vacíos. Para indicar qué valor se debe considerar "vacío", establezca este valor en la propiedad PLOT_EMPTY_VALUE. Por ejemplo, si un indicador debe dibujarse con segmentos sobre los valores no nulos, entonces hay que establecer el valor nulo como vacío:

```
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetDouble(índice_ de_construcción_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

Rellene siempre todos los elementos del búfer indicador con valores de forma explícita, estableciendo el valor vacío para los elementos que no van a dibujarse.

El número de búfers requeridos para construir DRAW_SECTION — 1.

Aquí tenemos un ejemplo del indicador que traza segmentos entre los precios High y Low. El color, grosor y el estilo de **todos** los segmentos se cambian de forma aleatoria cada **N** tics.



Fíjense que inicialmente para la construcción gráfica **plot1** con el estilo DRAW_SECTION las propiedades se establecen mediante la directiva del compilador #property, y luego en la función OnCalculate() estas tres propiedades se establecen de forma aleatoria. El parámetro N está pasado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                                DRAW_SECTION.mq5 |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_SECTION"
#property description "Dibujamos con segmentos rectos dentro de cada bars barras"
#property description "El color, grosor y el estilo del segmento se cambia de forma al
#property description "dentro de cada N tics"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Section
#property indicator_label1  "Section"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parámetro input
input int      bars=5;           // longitud del segmento en barras
input int      N=5;              // número de tics para el cambio del estilo de segmer
//--- búfer indicador para la construcción
double         SectionBuffer[];
//--- una variable auxiliar para calcular los extremos de segmentos
int            divider;
//--- array para almacenar colores
color colors[]={clrRed,clrBlue,clrGreen};
//--- array para almacenar estilos de trazado de la línea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array con el búfer indicador
   SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- comprobaremos el parámetro del indicador
   if(bars<=0)
     {
      PrintFormat("Valor del parámetro bar=%d inválido",bars);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=2*bars;
//---+
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número crítico de tics,
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- número de la barra a partir del cual empezaremos a calcular los valores del indi
   int start=0;
//--- si el indicador ha sido calculado antes, estableceremos start para la barra ante
   if(prev_calculated>0) start=prev_calculated-1;
//--- aquí están todos los cálculos de los valores del indicador
   for(int i=start;i<rates_total;i++)
     {
      //--- obtendremos el remanente de la división del número de la barra por 2*bars
      int rest=i%divider;
      //--- si el número de la barra se divide por 2*bars sin remanente
      if(rest==0)
        {
         //--- estableceremos el extremo del segmento en el precio High de esta barra
         SectionBuffer[i]=high[i];
        }
      //--- si la remanente de la división es igual a bars,
      else
        {
         //--- estableceremos el extremo del segmento en el precio High de esta barra
         if(rest==bars) SectionBuffer[i]=low[i];
         //--- si no encaja nada, omitimos esta barra - ponemos el valor 0
```

© 2000-2025, MetaQuotes Ltd.

```
         else SectionBuffer[i]=0;
        }
      }
//--- volveremos el valor prev_calculated para la siguiente llamada de la función
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| cambia la apariencia del segmento en el indicador                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de la línea
   string comm="";
//--- bloque del cambio del color de la línea
   int number=MathRand(); // obtenemos un número aleatorio
//--- el divisor del número es igual al tamaño del array colors[]
   int size=ArraySize(colors);
//--- obtenemos el índice para seleccionar nuevo color como el remanente de la divisió
   int color_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- apuntaremos el color de la línea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- bloque de cambio del grosor de la línea
   number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5;   // el grosor puede ser de 0 a 4
//--- fijaremos el grosor
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- bloque de cambio del estilo de la línea
   number=MathRand();
//--- el divisor del número es igual al tamaño del array styles
   size=ArraySize(styles);
//--- obtenemos el índice para seleccionar nuevo estilo como el remanente de la divisi
   int style_index=number%size;
//--- estableceremos el estilo de la línea
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- apuntaremos el estilo de la línea
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

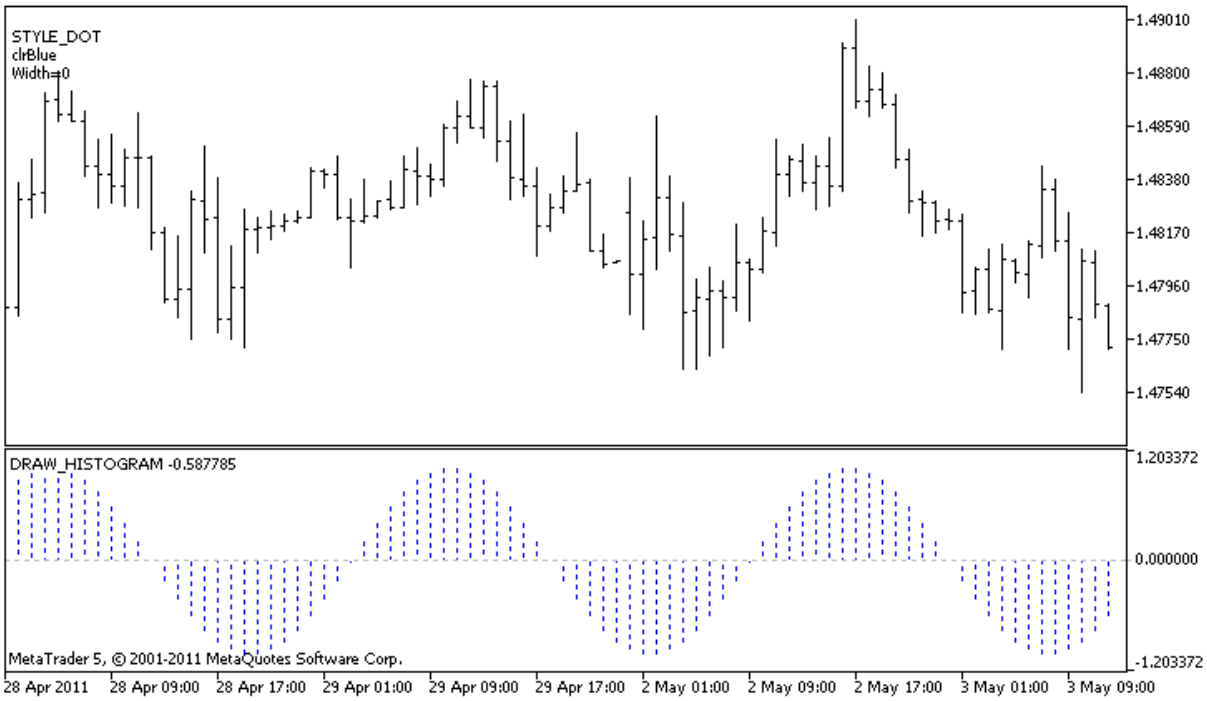
DRAW_HISTOGRAM

El estilo DRAW_HISTOGRAM dibuja un histograma como una secuencia de columnas de color especificado desde cero hasta el valor especificado . Los valores se cogen desde el búfer indicador. El grosor, color y el estilo de representación de la columna se puede establecer de la misma manera como para el estilo DRAW_LINE - con las directivas del compilador, o bien dinámicamente, utilizando la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite cambiar la apariencia del histograma en función de la situación actual.

Ya que en cada barra se dibuja una columna desde el nivel cero, es mejor utilizar DRAW_HISTOGRAM para mostrar en otra subventana del gráfico. Las más de las veces este tipo de construcción gráfica se utiliza para crear los indicadores del tipo oscilatorio, por ejemplo: Bears Power o OsMA. Para los valores vacíos que no se muestran, será suficiente indicarlos valores nulos.

El número de búfers requeridos para construir DRAW_HISTOGRAM — 1.

Aquí tenemos un ejemplo del indicador que dibuja una sinusoide de color especificado basándose en la función MathSin(). El color, grosor y el estilo de **todas** las columnas se cambian de forma aleatoria cada **N** tics. El parámetro bars determina el período de la sinusoide, eso quiere decir que dentro de una cantidad de barras especificada la sinusoide va a repetir su ciclo.



Fíjense que inicialmente para la construcción gráfica **plot1** con el estilo DRAW_HISTOGRAM las propiedades se establecen mediante la directiva del compilador #property, y luego en la función OnCalculate() estas tres propiedades se establecen de forma aleatoria. El parámetro N está pasado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_HISTOGRAM"
#property description "Dibuja la sinusoide como un histograma en otra ventana"
#property description "El color y el grosor de las columnas se cambian de forma aleato
#property description "dentro de cada N tics"
#property description "El parámetro bars establece el número de barras en el ciclo de

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parámetros input
input int      bars=30;          // período de la sinusoide en barras
input int      N=5;              // número de tics para el cambio del histograma
//--- indicator buffers
double         HistogramBuffer[];
//--- factor para obtener el ángulo 2Pi en radiánes al multiplicar por el parámetro ba
double    multiplier;
//--- array para almacenar colores
color colors[]={clrRed,clrBlue,clrGreen};
//--- array para almacenar estilos de trazado de la línea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- calcularemos el multiplicador
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Establezca el valor bars=%d mayor que 1",bars);
      //--- finalización anticipada del trabajo del indicador
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número crítico de tics,
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- cálculos de los valores del indicador
   int start=0;
//--- si el cálculo ya ha sido realizado en el arranque anterior de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // fijaremos el inicio del cálculo a
//--- llenamos el búfer indicador con valores
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=sin(i*multiplier);
     }
//--- volveremos el valor prev_calculated para la siguiente llamada de la función
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de la línea en el indicador                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de la línea
   string comm="";
//--- bloque de cambio del color de la línea
   int number=MathRand(); // obtenemos un número aleatorio
//--- el divisor del número es igual al tamaño del array colors[]
   int size=ArraySize(colors);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- obtenemos el índice para seleccionar nuevo color como el remanente de la divisió
   int color_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- apuntaremos el color de la línea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- bloque de cambio del grosor de la línea
   number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5;   // el grosor puede ser de 0 a 4
//--- fijaremos el grosor
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- bloque de cambio del estilo de la línea
   number=MathRand();
//--- el divisor del número es igual al tamaño del array styles
   size=ArraySize(styles);
//--- obtenemos el índice para seleccionar nuevo estilo como el remanente de la divisi
   int style_index=number%size;
//--- estableceremos el estilo de la línea
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- apuntaremos el estilo de la línea
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_HISTOGRAM2

El estilo DRAW_HISTOGRAM2 dibuja un histograma de color especificado - segmentos verticales, usando valores de dos búfers indicadores. El grosor, color y el estilo de segmentos se puede establecer de la misma manera como para el estilo DRAW_LINE - con las directivas del compilador, o bien dinámicamente, utilizando la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite cambiar la apariencia del histograma en función de la situación actual.

El estilo DRAW_HISTOGRAM se puede utilizar tanto en una subventana separada del gráfico, como en la ventana principal. Los valores vacíos no se dibujan, todos los valores hay que establecer en los búfers indicadores de forma explícita. Los búfers no se inicializan con valores vacíos.

El número de búfers requeridos para construir DRAW_HISTOGRAM2 — 2.

Aquí tenemos un ejemplo del indicador que traza en cada barra un segmento vertical de color y grosor especificados entre los precios Open y Close. El color, grosor y el estilo de **todas** las columnas del histograma se cambian de forma aleatoria cada **N** tics. Cuando el indicador se inicia, en la función OnInit() se define al azar el número del día de la semana para el que el histograma no va a dibujarse - invisible_day. Para eso se establece el valor vacío PLOT_EMPTY_VALUE=0:

```
//--- estableceremos el valor vacío
   PlotIndexSetDouble(índice_ de_construcción_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```



Fíjense que inicialmente para la construcción gráfica **plot1** con el estilo DRAW_HISTOGRAM2 las propiedades se establecen mediante la directiva del compilador #property, y luego en la función OnCalculate() estas tres propiedades se establecen de forma aleatoria. El parámetro N está pasado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM2.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_HISTOGRAM2"
#property description "Dibuja en cada barra un segmento entre Open y Close"
#property description "El color, grosor y el estilo se cambian de forma aleatoria"
#property description "dentro de cada N tics"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot Histogram_2
#property indicator_label1  "Histogram_2"
#property indicator_type1   DRAW_HISTOGRAM2
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      N=5;              // número de tics para el cambio del histograma
//--- indicator buffers
double         Histogram_2Buffer1[];
double         Histogram_2Buffer2[];
//--- día de la semana para el que el indicador no se dibuja
int invisible_day;
//--- array para almacenar colores
color colors[]={clrRed,clrBlue,clrGreen};
//--- array para almacenar estilos de trazado de la línea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//--- estableceremos el valor vacío
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- obtenemos un número aleatorio de 0 a 5
   invisible_day=MathRand()%6;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número crítico de tics,
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- cálculos de los valores del indicador
   int start=0;
//--- para obtener el día de la semana por la hora de apertura de cada barra
   MqlDateTime dt;
//--- si el cálculo ya ha sido realizado en el arranque anterior de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // fijaremos el inicio del cálculo a
//--- llenamos el búfer indicador con valores
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      if(dt.day_of_week==invisible_day)
        {
         Histogram_2Buffer1[i]=0;
         Histogram_2Buffer2[i]=0;
        }
      else
        {
         Histogram_2Buffer1[i]=open[i];
         Histogram_2Buffer2[i]=close[i];
        }
     }
//--- volveremos el valor prev_calculated para la siguiente llamada de la función
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de la línea en el indicador                 |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de la línea
   string comm="";
//--- bloque del cambio del color de la línea
   int number=MathRand(); // obtenemos un número aleatorio
//--- el divisor del número es igual al tamaño del array colors[]
   int size=ArraySize(colors);
//--- obtenemos el índice para seleccionar nuevo color como el remanente de la divisió
   int color_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- apuntaremos el color de la línea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- bloque de cambio del grosor de la línea
   number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5;   // el grosor puede ser de 0 a 4
//--- fijaremos el grosor de la línea
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- bloque de cambio del estilo de la línea
   number=MathRand();
//--- el divisor del número es igual al tamaño del array styles
   size=ArraySize(styles);
//--- obtenemos el índice para seleccionar nuevo estilo como el remanente de la divisi
   int style_index=number%size;
//--- estableceremos el estilo de la línea
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- apuntaremos el estilo de la línea
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- agregaremos la información sobre el día que se ignora en los cálculos
   comm="\r\nDía no dibujable - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_ARROW

El estilo DRAW_ARROW dibuja en el gráfico las flechas de color especificado (símbolos del conjunto Wingdings) basándose en el valor del búfer indicador. El grosor y el color de los símbolo se puede establecer de la misma manera como para el estilo DRAW_LINE - con las directivas del compilador, o bien dinámicamente, utilizando la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite cambiar la apariencia del indicador en función de la situación actual.

El código del símbolo a mostrar en el gráfico se establece a través de la propiedad PLOT_ARROW.

```
//--- estableceremos el código del símbolo desde el fuente Wingdings para dibujar en
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

Por defecto, el valor de PLOT_ARROW=159 (un círculo).

Cada flecha prácticamente es un símbolo que tiene su alto y su punto de enlace, y puede cubrir alguna información importante en el gráfico (por ejemplo, el precio del cierre en la barra). Por eso se puede indicar adicionalmente el desplazamiento vertical en píxeles, que no depende de la escala del gráfico. Las flechas se desplazarán visualmente por la línea vertical a esta especificada cantidad de píxeles, aunque los valores del indicador se quedarán los mismos:

```
//--- estableceremos el desplazamiento de flechas por la línea vertical en píxeles
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

Un valor negativo de PLOT_ARROW_SHIFT significa el desplazamiento de las flechas hacia arriba, un valor positivo significa el desplazamiento de la flecha hacia abajo.

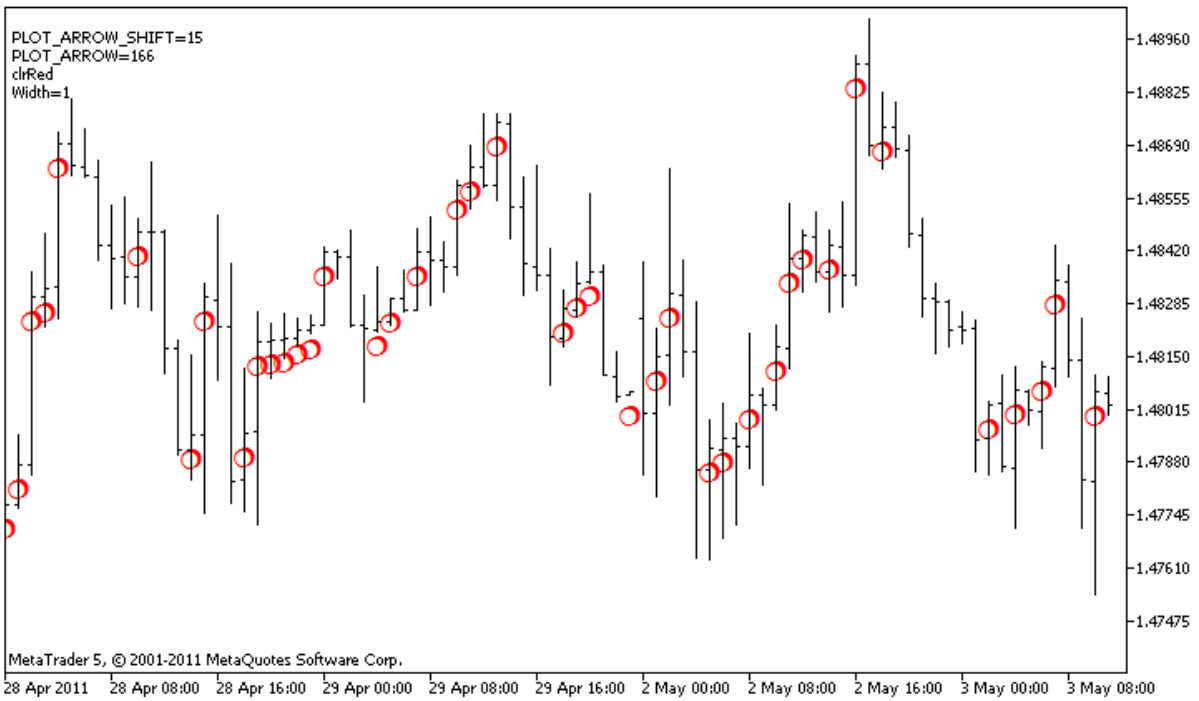
El estilo DRAW_ARROW se puede utilizar tanto en una subventana separada del gráfico, como en la ventana principal. Los valores vacíos no se dibujan y no se muestran en la "Ventana de datos", todos los valores hay que establecer en los búfers indicadores de forma explícita. Los búfers no se inicializan con valores vacíos.

```
//--- estableceremos el valor vacío
   PlotIndexSetDouble(índice_ de_construcción_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

El número de búfers requeridos para construir DRAW_ARROW — 1.

Aquí tenemos un ejemplo del indicador que dibuja las flechas en cada barra cuyo precio de cierre Close es más alto que el precio de cierre de la barra anterior. El color, grosor, desplazamiento y el código del símbolo de **todas** las flechas se cambian de forma aleatoria cada **N** tics.

© 2000-2025, MetaQuotes Ltd.



En el ejemplo, inicialmente para la construcción gráfica **plot1** con el estilo DRAW_ARROW las propiedades del color y el tamaño se establecen mediante la directiva del compilador #property, y luego en la función OnCalculate() las propiedades se establecen de forma aleatoria. El parámetro N está pasado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_ARROW.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_ARROW"
#property description "Dibuja las flechas determinadas por los caracteres de Unicode e
#property description "El color, tamaño, desplazamiento y el código del símbolo de la
#property description "dentro de cada N tics"
#property description "El parámetro code establece el valor base: código=159 (círculo)

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Arrows
#property indicator_label1  "Arrows"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrGreen
#property indicator_width1  1
//--- parámetros input
```

```
input int      N=5;          // número de tics para el cambio
input ushort   code=159;     // código del símbolo a dibujar en DRAW_ARROW
//--- búfer indicador para la construcción
double         ArrowsBuffer[];
//--- array para almacenar colores
color colors[]={clrRed,clrBlue,clrGreen};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- estableceremos el código del símbolo para dibujar en PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- estableceremos el desplazamiento de flechas por la línea vertical en píxeles
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- estableceremos un 0 como valor vacío
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del color, tamaño, desplazamiento y código de l
   ticks++;
//--- si tenemos acumulado un número crítico de tics,
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- bloque para calcular los valores del indicador
```

© 2000-2025, MetaQuotes Ltd.

```
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- ciclo del cálculo
   for(int i=1;i<rates_total;i++)
     {
      //--- si el precio actual Close es más alto que el anterior, colocamos la flecha
      if(close[i]>close[i-1])
         ArrowsBuffer[i]=close[i];
      //--- en caso contrario, mostramos el valor cero
      else
         ArrowsBuffer[i]=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de símbolos en el indicador                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades del indicador
   string comm="";
//--- bloque del cambio del color de la flecha
   int number=MathRand(); // obtenemos un número aleatorio
//--- el divisor del número es igual al tamaño del array colors[]
   int size=ArraySize(colors);
//--- obtenemos el índice para seleccionar nuevo color como el remanente de la divisió
   int color_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- apuntaremos el color de la línea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- bloque del cambio del tamaño de flechas
   number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5;   // el tamaño puede ser de 0 a 4
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el tamaño de flechas
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- bloque del cambio del código de la flecha (PLOT_ARROW)
   number=MathRand();
//--- obtendremos el remanente de la división de números enteros para calcular nuevo c
   int code_add=number%20;
//--- estableceremos el nuevo código del símbolo como la suma code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- apuntaremos el código del símbolo PLOT_ARROW
```

© 2000-2025, MetaQuotes Ltd.

```
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- bloque del cambio del desplazamiento de flechas por la línea vertical en píxeles
   number=MathRand();
//--- obtenemos el desplazamiento como el remanente de la división de números enteros
   int shift=20-number%41;
//--- estableceremos un nuevo desplazamiento de -20 a 20
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- apuntaremos el desplazamiento PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_ZIGZAG

El estilo DRAW_ZIGZAG dibuja segmentos de color especificado, usando valores de dos búfers indicadores. Este estilo es muy parecido al DRAW_SECTION, pero a diferencia del último, éste permite dibujar segmentos verticales dentro de los márgenes de una barra, si los valores de los dos búfers indicadores están establecidos para esta barra. Los segmentos se trazan desde un valor en el primer búfer indicador hasta un valor en el segundo. Ninguno de los dos búferes puede contener sólo valores vacíos. Si es así, no se dibuja nada.

El grosor, color y el estilo de la línea se puede establecer de la misma manera como para el estilo DRAW_SECTION - con las directivas del compilador, o bien dinámicamente, utilizando la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite "vivificar" los indicadores, para que cambien su apariencia en función de la situación actual.

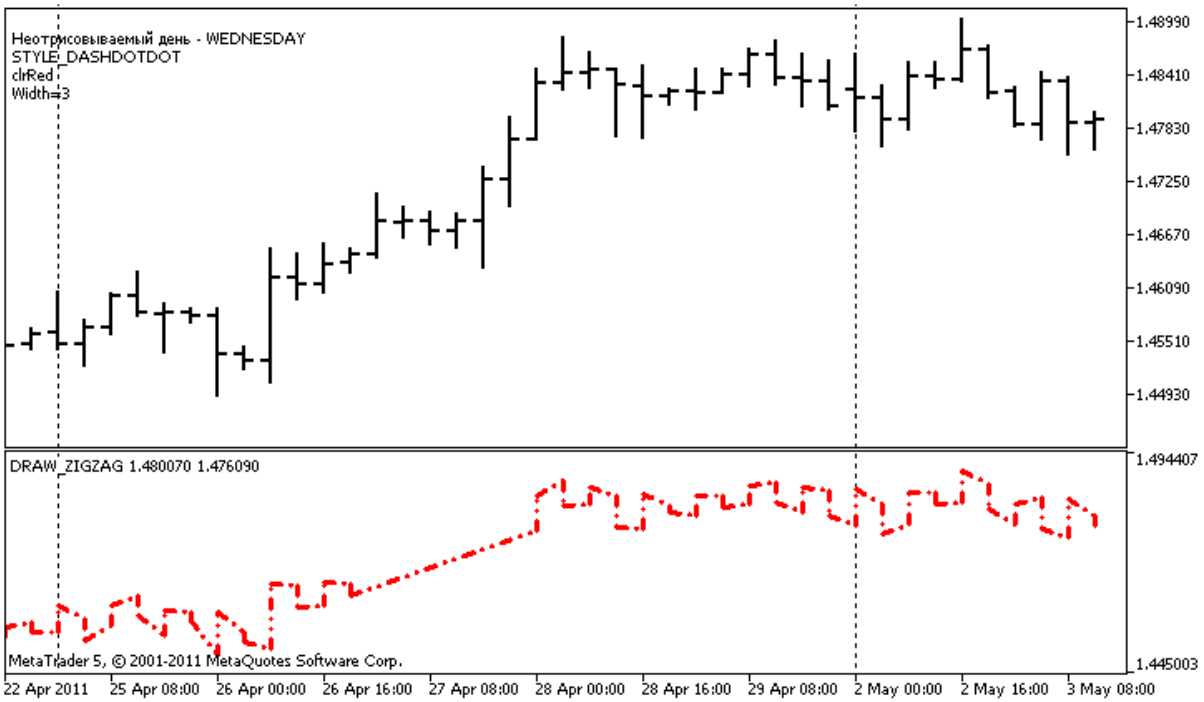
Los segmentos se trazan desde un valor no vacío de un búfer hasta otro valor no vacío de otro búfer indicador. Para indicar qué valor se debe considerar "vacío", establezca este valor en la propiedad PLOT_EMPTY_VALUE:

```
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetDouble(índice_ de_construcción_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Rellene siempre los búfers indicadores con valores de forma explícita, para las barras que se ignoran indique en el búfer un valor vacío.

El número de búfers requeridos para construir DRAW_ZIGZAG — 2.

Aquí tenemos un ejemplo del indicador que traza la sierra a base de los precios High y Low. El color, grosor y el estilo de la línea del zigzag se cambian de forma aleatoria cada **N** tics.



Fíjense en que inicialmente para la construcción gráfica **plot1** con el estilo DRAW_ZIGZAG las propiedades se establecen mediante la directiva del compilador #property, y luego en la función OnCalculate() estas tres propiedades se establecen de forma aleatoria. El parámetro N está pasado a

© 2000-2025, MetaQuotes Ltd.

los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                                  DRAW_ZIGZAG.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_ZIGZAG"
#property description "Dibuja con segmentos rectos \"la sierra\", saltando las barras
#property description "El día que se salta se elige de forma aleatoria al iniciar el i
#property description "El color, grosor y el estilo de segmentos se cambia de forma al
#property description " dentro de cada N tics"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ZigZag
#property indicator_label1  "ZigZag"
#property indicator_type1   DRAW_ZIGZAG
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parámetros input
input int      N=5;              // número de tics a cambiar
//--- indicator buffers
double         ZigZagBuffer1[];
double         ZigZagBuffer2[];
//--- día de la semana para el que el indicador no se dibuja
int invisible_day;
//--- array para almacenar colores
color colors[]={clrRed,clrBlue,clrGreen};
//--- array para almacenar estilos de trazado de la línea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de arrays con búfers indicadores
   SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- obtenemos un número aleatorio de 0 a 6, para este día el indicador no se dibuja
   invisible_day=MathRand()%6;
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número suficiente de tics
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- necesitaremos una estructura de tiempo para obtener el día de la semana de cada
   MqlDateTime dt;

//--- posición de inicio del cálculo
   int start=0;
//--- si el indicador se calcula en el tic anterior, empezamos el cálculo a partir del
   if(prev_calculated!=0) start=prev_calculated-1;
//--- ciclo de cálculos
   for(int i=start;i<rates_total;i++)
     {
      //--- apuntaremos en la estructura la hora de apertura de la barra
      TimeToStruct(time[i],dt);
      //--- si el día de la semana de esta barra es igual a invisible_day
      if(dt.day_of_week==invisible_day)
        {
         //--- apuntaremos en el búfer los valores vacíos para esta barra
         ZigZagBuffer1[i]=0;
```

© 2000-2025, MetaQuotes Ltd.

```
         ZigZagBuffer2[i]=0;
        }
      //--- si el día de la semana es correcto, llenamos los búfers
      else
        {
         //--- si el número de la barra es par
         if(i%2==0)
           {
            //---  escribimos en el 1-r búfer High, en el 2-do Low
            ZigZagBuffer1[i]=high[i];
            ZigZagBuffer2[i]=low[i];
           }
         //--- número de la barra impar
         else
           {
            //--- llenamos la barra en sentido inverso
            ZigZagBuffer1[i]=low[i];
            ZigZagBuffer2[i]=high[i];
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de segmentos en el zigzag                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades del ZigZag
   string comm="";
//--- bloque del cambio del color del zigzag
   int number=MathRand(); // obtenemos un número aleatorio
//--- el divisor del número es igual al tamaño del array colors[]
   int size=ArraySize(colors);
//--- obtenemos el índice para seleccionar nuevo color como el remanente de la divisió
   int color_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- apuntaremos el color de la línea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- bloque de cambio del grosor de la línea
   number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5;   // el grosor puede ser de 0 a 4
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
```

© 2000-2025, MetaQuotes Ltd.

```
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- bloque de cambio del estilo de la línea
   number=MathRand();
//--- el divisor del número es igual al tamaño del array styles
   size=ArraySize(styles);
//--- obtenemos el índice para seleccionar nuevo estilo como el remanente de la divisi
   int style_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- apuntaremos el estilo de la línea
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- agregaremos la información sobre el día que se ignora en los cálculos
   comm="\r\nDía no dibujable - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_FILLING

El estilo DRAW_FILLING dibuja un área de color entre los valores de dos búferes de indicadores. Prácticamente, este estilo traza dos líneas y colorea el espacio entre ellas en uno de dos colores predefinidos. Sirve para crear indicadores que dibujan canales. Ninguno de los dos búferes puede contener sólo valores vacíos. Si es así, no se dibuja nada.

Se puede definir dos colores para el relleno:

- el primer color se utiliza para las áreas donde los valores en el primer búfer indicador son más altos que los valores en el segundo búfer indicador;
- el segundo color se utiliza para las áreas donde los valores en el segundo búfer indicador son más altos que los valores en el primer búfer indicador.

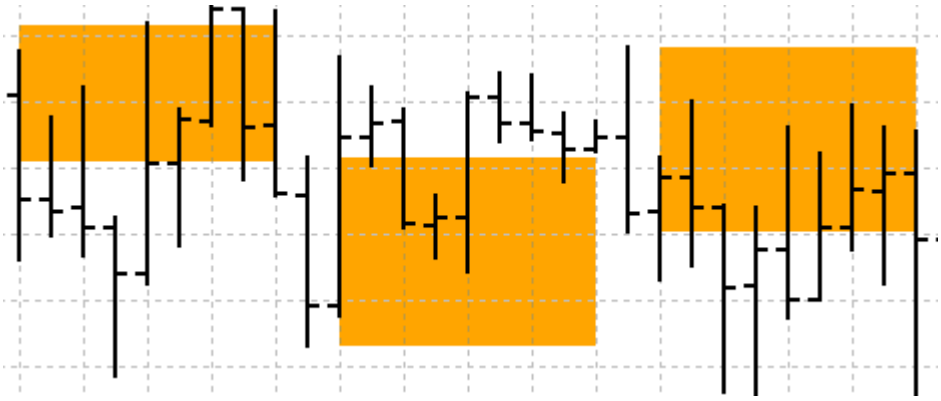
El color de relleno se puede definir con las directivas del compilador, o dinámicamente a través de la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite "vivificar" los indicadores, para que cambien su apariencia en función de la situación actual.

El indicador se calcula para todas las barras para las que los valores de ambos búferes de indicadores no son iguales a cero y no son iguales al valor vacío. Para indicar qué valor habrá que considerar como "vacío", especifíquelo en la propiedad PLOT_EMPTY_VALUE:

```
   #define INDICATOR_EMPTY_VALUE -1.0
   ...
//--- el valor INDICATOR_EMPTY_VALUE (valor vacío) no va a participar en el cálculo
   PlotIndexSetDouble(índice_de_construcción_DRAW_FILLING,PLOT_EMPTY_VALUE,INDICATOR_E
```

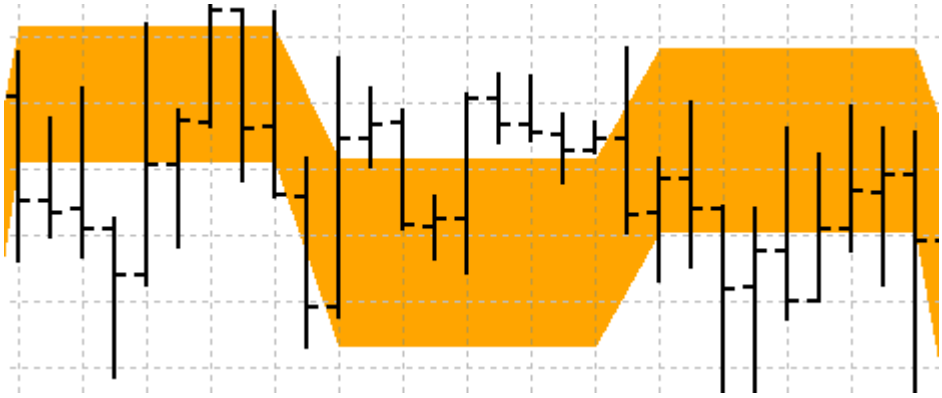
El dibujo sobre las barras que no participan en el cálculo del indicador va a depender de los valores situados en los búferes de indicadores:

- Las barras para las que los valores de ambos búferes de indicadores son iguales a 0 no participan en el dibujo del indicador. Es decir el área con los valores iguales a cero no va a colorearse.



- Las barras para las que los valores de los búferes de indicadores son iguales a "valor vacío" participan en el dibujo del indicador. El área con valores vacíos va a colorearse de tal manera que las zonas con valores significativos se unan.

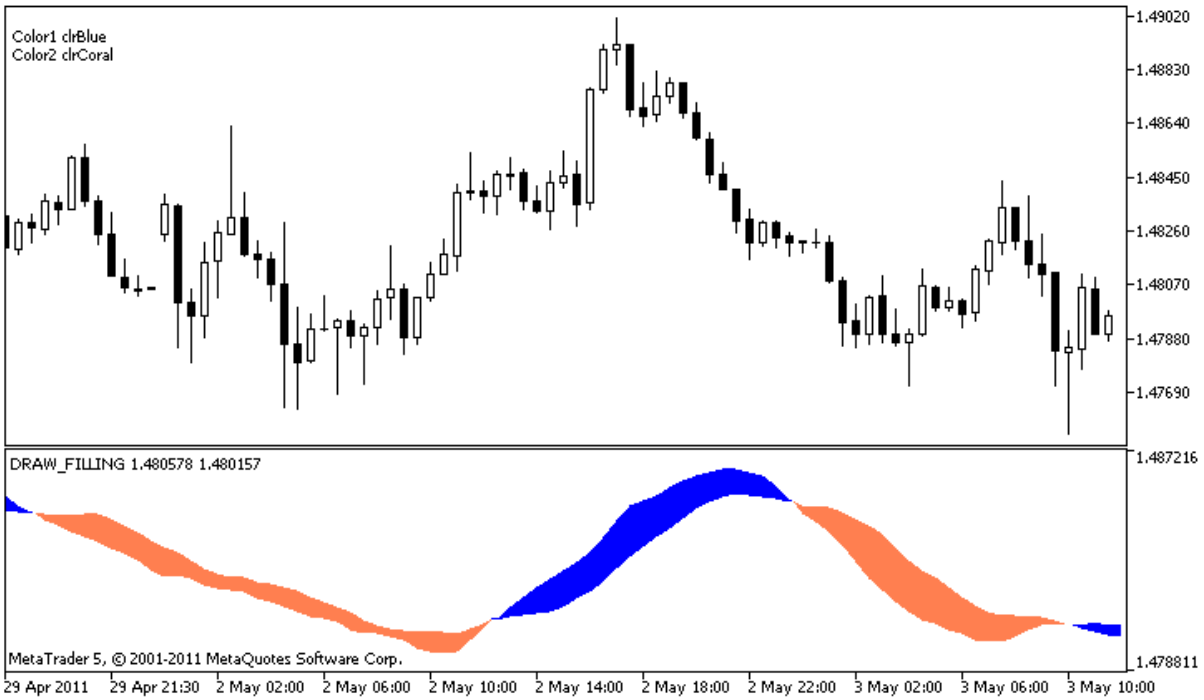
© 2000-2025, MetaQuotes Ltd.



Es importante mencionar que si el "valor vacío" es igual a cero, las barras que no participan en el cálculo del indicador también van a colorearse.

El número de búfers requeridos para construir DRAW_FILLING — 2.

Aquí tenemos un ejemplo del indicador que dibuja en una ventana separada un canal entre dos medias móviles con diferentes períodos de promedio. El cambio de color durante el cruce de las medias muestra visualmente el cambio de la tendencia alcista y bajista. Los colores se cambian aleatoriamente cada **N** tics. El parámetro N está sacado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).



Fíjense en que inicialmente para la construcción gráfica **plot1** con el estilo DRAW_FILLING dos colores se establecen mediante la directiva del compilador #property, y luego en la función OnCalculate() nuevos colores se establecen de forma aleatoria.

```
//+------------------------------------------------------------------+
//|                                                DRAW_FILLING.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_FILLING"
#property description "Dibuja en la ventana separada un canal entre dos medias móviles
#property description "El color del relleno del canal se cambia de forma aleatoria"
#property description "dentro de cada N tics"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot Intersection
#property indicator_label1  "Intersection"
#property indicator_type1   DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1  1
//--- parámetros input
input int      Fast=13;          // período de la media móvil rápida
input int      Slow=21;          // período de la media móvil lenta
input int      shift=1;          // desplazamiento de las medias móviles hacia el futu
input int      N=5;              // número de tics a cambiar
//--- búfers indicadores
double         IntersectionBuffer1[];
double         IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- array para almacenar colores
color colors[]={clrRed,clrBlue,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrCo
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,IntersectionBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,IntersectionBuffer2,INDICATOR_DATA);
//---
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//---
   fast_handle=iMA(_Symbol,_Period,Fast,0,MODE_SMA,PRICE_CLOSE);
   slow_handle=iMA(_Symbol,_Period,Slow,0,MODE_SMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número suficiente de tics
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- hacemos el primer cálculo del indicador, o los datos han cambiado y se requiere
   if(prev_calculated==0)
     {
      //--- copiamos todos los valores de los indicadores a los búfers correspondiente
      int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
     }
   else // llenamos sólo aquellos datos que se han actualizado
     {
      //--- obtendremos la diferencia en barras entre el arranque actual y el anterior
      int to_copy=rates_total-prev_calculated;
      //--- si no hay diferencia, igualmente copiaremos un valor - en la barra cero
      if(to_copy==0) to_copy=1;
      //--- copiamos to_copy valores al mismísimo final de los búfers indicadores
      int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Cambia los colores del relleno del canal                         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de la línea
```

© 2000-2025, MetaQuotes Ltd.

```
   string comm="";
//--- bloque de cambio del color de la línea
   int number=MathRand(); // obtenemos un número aleatorio
//--- el divisor del número es igual al tamaño del array colors[]
   int size=ArraySize(colors);

//--- obtenemos el índice para seleccionar nuevo color como el remanente de la divisió
   int color_index1=number%size;
//--- establecemos el primer color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
//--- apuntaremos el primer color
   comm=comm+"\r\nColor1 "+(string)colors[color_index1];

//--- obtenemos el índice para seleccionar nuevo color como el remanente de la divisió
   number=MathRand(); // obtenemos un número aleatorio
   int color_index2=number%size;
//--- establecemos el segundo color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
//--- apuntaremos el segundo color
   comm=comm+"\r\nColor2 "+(string)colors[color_index2];
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_BARS

El estilo DRAW_BARS dibuja las barras basándose en los valores de cuatro búfers indicadores que contienen los precios Open, High, Low y Close. Se utiliza para crear sus propios indicadores personalizados en forma de barras, también en otra subventana del gráfico y para otros instrumentos financieros.

El color de las barras se puede definir con las directivas del compilador, o dinámicamente a través de la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite "vivificar" los indicadores, para que cambien su apariencia en función de la situación actual.

El indicador se dibuja sólo para las barras que tienen establecidos los valores no vacíos de **todos** los cuatro búfers indicadores. Para indicar qué valor se debe considerar "vacío", establezca este valor en la propiedad PLOT_EMPTY_VALUE:

```
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetDouble(índice_ de_construcción_DRAW_BARS,PLOT_EMPTY_VALUE,0);
```

Rellene siempre los búfers indicadores con valores de forma explícita, para las barras que se ignoran indique en el búfer un valor vacío.

El número de búfers requeridos para construir DRAW_BARS — 4. Todos los búfers que se utilizan para la construcción deben ir en serie uno detrás del otro en orden establecido: Open, High, Low y Close. Ninguno de los búfers puede contener sólo los valores vacíos, porque si es así, no se dibuja nada.

Aquí tenemos un ejemplo del indicador que dibuja en una ventana separada las barras para el instrumento financiero especificado. El color de las barras se cambia aleatoriamente cada **N** tics. El parámetro N está sacado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).



© 2000-2025, MetaQuotes Ltd.

Fíjense en que inicialmente para la construcción gráfica **plot1** con el estilo DRAW_BARS el color se establece mediante la directiva del compilador #property, y luego en la función OnCalculate() el color se elige de forma aleatoria.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_BARS.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_BARS"
#property description "Dibuja en la ventana separada las barras para el símbolo seleco
#property description "El color y el grosor de las barras, igual que el estilo, se car
#property description "cada N tics"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "Bars"
#property indicator_type1   DRAW_BARS
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parámetros input
input int      N=5;              // número de tics para el cambio de apariencia
input int      bars=500;         // número de barras a mostrar
input bool     messages=false;   // mostrar mensajes en el log "Asesores Expertos"
//--- búfers indicadores
double         BarsBuffer1[];
double         BarsBuffer2[];
double         BarsBuffer3[];
double         BarsBuffer4[];
//--- nombre del símbolo
string symbol;
//--- array para almacenar colores
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- si bars es muy pequeño - finalizamos el trabajo antes de tiempo
   if(bars<50)
     {
      Comment("¡Por favor, indique el número más grande de barras! El trabajo del indi
```

© 2000-2025, MetaQuotes Ltd.

```
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
//--- nombre del símbolo para el que se dibujan las barras
   symbol=_Symbol;
//--- estableceremos la visualización del símbolo
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- valor vacío
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número suficiente de tics
   if(ticks>=N)
     {
      //--- escogeremos nuevo símbolo en la ventana "Observación del mercado"
      symbol=GetRandomSymbolName();
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();

      int tries=0;
      //--- haremos 5 intentos de llenar el búfer con los precios desde symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
         //--- contador de llamadas a la función CopyFromSymbolToBuffers()
         tries++;
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  Llenamos los búfers indicadores con los precios                 |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total)
  {
//--- vamos a copiar los precios Open, High, Low y Close al array rates[]
   MqlRates rates[];
//--- contador de intentos
   int attempts=0;
//--- cantidad que se ha copiado ya
   int copied=0;
//--- hacemos 25 intentos de obtener la serie temporal para el símbolo necesario
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si no se ha podido copiar la cantidad suficiente de barras
   if(copied!=bars)
     {
      //--- formaremos la cadena del mensaje
      string comm=StringFormat("Para el símbolo %s se ha logrado obtener sólo %d barra
                               name,
                               copied,
                               bars
                               );
      //--- mostraremos el mensaje en un comentario en la ventana principal del gráfic
      Comment(comm);
      //--- mostramos el mensaje
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- estableceremos la visualización del símbolo
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+name+")");
     }
//--- inicializaremos los búfers con valores vacíos
   ArrayInitialize(BarsBuffer1,0.0);
   ArrayInitialize(BarsBuffer2,0.0);
   ArrayInitialize(BarsBuffer3,0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayInitialize(BarsBuffer4,0.0);
//--- copiamos los precios a los búfers
   for(int i=0;i<copied;i++)
     {
      //--- calcularemos el índice correspondiente para los búfers
      int buffer_index=total-copied+i;
      //--- escribimos los precios en los búfers
      BarsBuffer1[buffer_index]=rates[i].open;
      BarsBuffer2[buffer_index]=rates[i].high;
      BarsBuffer3[buffer_index]=rates[i].low;
      BarsBuffer4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| devuelve aleatoriamente el símbolo desde Market Watch            |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- número de símbolos mostrados en la ventana "Observación del mercado"
   int symbols=SymbolsTotal(true);
//--- posición del símbolo en la lista - un número aleatorio de 0 a symbols
   int number=MathRand()%symbols;
//--- devolvemos el nombre del símbolo sobre la posición indicada
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de las barras                               |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de las barras
   string comm="";
//--- bloque del cambio del color de las barras
   int number=MathRand(); // obtenemos un número aleatorio
//--- el divisor del número es igual al tamaño del array colors[]
   int size=ArraySize(colors);
//--- obtenemos el índice para seleccionar nuevo color como el remanente de la divisi
   int color_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- apuntaremos el color de la línea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- bloque del cambio del grosor de las barras
   number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5;   // el grosor puede ser de 0 a 4
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- apuntamos el nombre del símbolo
   comm="\r\n"+symbol+comm;

//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_CANDLES

El estilo DRAW_CANDLES dibuja las velas japonesas basándose en los valores de cuatro búfers indicadores que contienen los precios Open, High, Low y Close. Se utiliza para crear sus propios indicadores personalizados en forma de velas japonesas, también en otra subventana del gráfico y para otros instrumentos financieros.

El color de las velas se puede definir usando las directivas del compilador, o dinámicamente a través de la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite "vivificar" los indicadores, para que cambien su apariencia en función de la situación actual.

El indicador se dibuja sólo para las barras que tienen establecidos los valores no vacíos de **todos** los cuatro búfers indicadores. Para indicar qué valor se debe considerar "vacío", establezca este valor en la propiedad PLOT_EMPTY_VALUE:

```
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetDouble(índice_ de_construcción_DRAW_CANDLES,PLOT_EMPTY_VALUE,0);
```

Rellene siempre los búfers indicadores con valores de forma explícita, para las barras que se ignoran indique en el búfer un valor vacío.

El número de búfers requeridos para construir DRAW_CANDLES — 4. Todos los búfers que se utilizan para la construcción deben ir en serie uno detrás del otro en orden establecido: Open, High, Low y Close. Ninguno de los búfers puede contener sólo los valores vacíos, porque si es así, no se dibuja nada.

Para el estilo DRAW_CANDLES es posible indicar de uno a tres colores, dependiendo de ello, cambiará el aspecto exterior de las velas. Si se indica solo un color, entonces todas las velas del gráfico se colorearán por entero de ese color.

```
//--- velas iguales, coloreadas en un solo color
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- solo se ha indicado un color, por eso todas las velas serán de un solo color
#property indicator_color1  clrGreen
```

Si indicamos dos colores a través de un guión, los contornos se dibujarán con el primer color, y el cuerpo con el segundo.

```
//--- el color de las velas se distingue del color de las sombras
#property indicator_label1  "Two color candles"
#property indicator_type1   DRAW_CANDLES
//--- las sombras y el contorno de las velas son de color verde, el cuerpo, de color b
#property indicator_color1  clrGreen,clrWhite
```

Para que sea posible mostrar las velas crecientes y decrecientes de forma distinta, es necesario indicar los tres colores con ayuda de comas. En este caso, el contorno de la vela se dibujará con el primer color, y el color de la vela alcista y bajista se establecerá con el segundo y el tercer color.

```
//--- el color de las velas se distingue del color de las sombras
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- las sombras y el contorno en color verde, el cuerpo de la vela alcista en color
```

© 2000-2025, MetaQuotes Ltd.

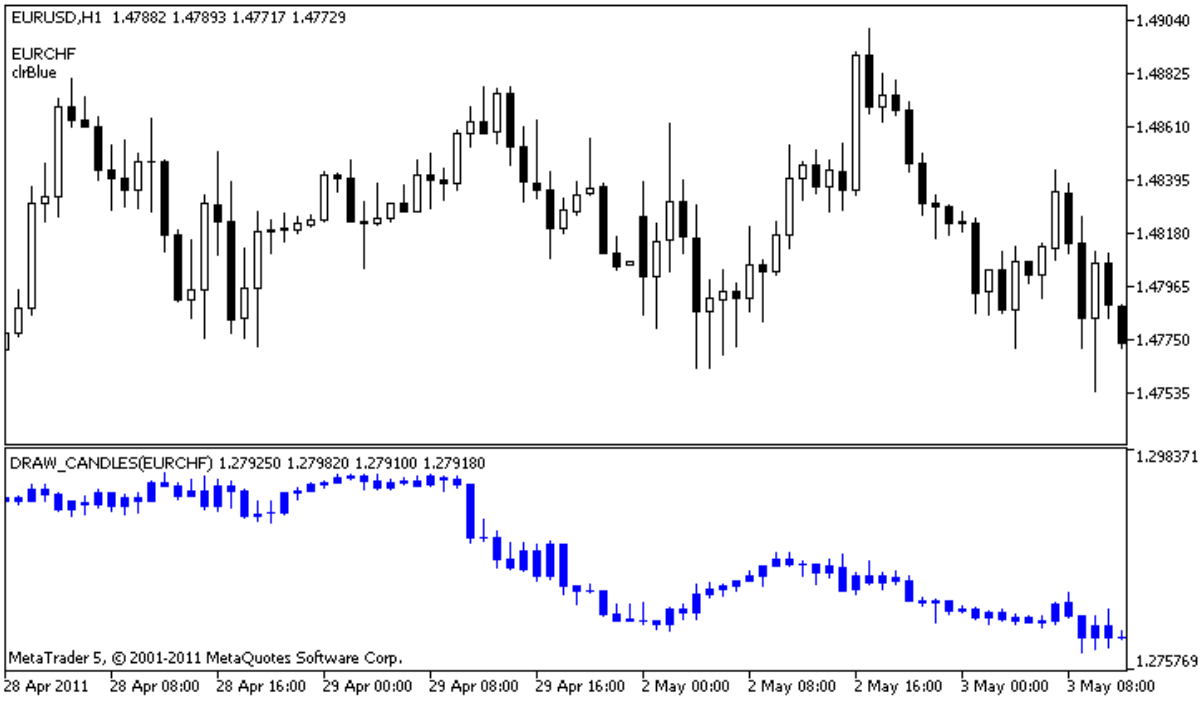
```
#property indicator_color1  clrGreen,clrWhite,clrRed
```

De esta forma, con la ayuda del estilo DRAW_CANDLES se pueden crear variantes personalizadas propias del color de las velas. Asimismo, todos los colores pueden cambiarse de forma dinámica durante el proceso del trabajo con la ayuda de la función PlotIndexSetInteger(índice_de_la_construcción_DRAW_CANDLES, PLOT_LINE_COLOR, número_del_modificador, color) , donde el número_del_modificador puede tener los valores siguientes:

- 0 - color del contorno y las sombras
- 1- color del cuerpo de la vela alcista
- 2 - color del cuerpo de vela bajista

```
//--- establecemos el color del contorno y de las sombras
PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrBlue);
//--- establecemos el color del cuerpo para la vela alcista
PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrGreen);
//--- establecemos el color del cuerpo para la vela bajista
PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrRed);
```

Aquí tenemos un ejemplo del indicador que dibuja en una ventana separada las velas japonesas para el instrumento financiero especificado. El color de las velas se cambia aleatoriamente cada **N** tics. El parámetro N está sacado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).



Fíjense en que inicialmente para la construcción gráfica **plot1** el color se establece mediante la directiva del compilador #property, y luego en la funciónOnCalculate() se elige aleatoriamente un nuevo color desde la lista previamente preparada.

```
//+------------------------------------------------------------------+
//|                                                  DRAW_CANDLES.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_CANDLES."
#property description "Dibuja en la ventana separada las velas para el símbolo selecci
#property description " "
#property description "El color y el grosor de las velas, igual que el estilo, se camb
#property description "de forma aleatoria cada N tics."

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "DRAW_CANDLES1"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- parámetros input
input int      N=5;              // número de tics para el cambio de apariencia
input int      bars=500;         // número de barras a mostrar
input bool     messages=false;   // mostrar mensajes en el log "Asesores Expertos"
//--- búfers indicadores
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- nombre del símbolo
string symbol;
//--- array para almacenar colores
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- si bars es muy pequeño - finalizamos el trabajo antes de tiempo
   if(bars<50)
     {
      Comment("¡Por favor, indique el número más grande de barras! El trabajo del indi
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
   SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
//--- valor vacío
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- nombre del símbolo para el que se dibujan las barras
   symbol=_Symbol;
//--- estableceremos la visualización del símbolo
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES("+symbol+")");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número suficiente de tics
   if(ticks>=N)
     {
      //--- escogeremos nuevo símbolo en la ventana "Observación del mercado"
      symbol=GetRandomSymbolName();
      //--- cambiamos la apariencia
      ChangeLineAppearance();
      //--- escogeremos nuevo símbolo en la ventana "Observación del mercado"
      int tries=0;
      //--- haremos 5 intentos de llenar el búfer plot1 con los precios desde symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
        {
         //--- contador de llamadas a la función CopyFromSymbolToBuffers()
         tries++;
        }
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  Llena la vela indicada                                          |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[]
                             )
  {
//--- vamos a copiar los precios Open, High, Low y Close al array rates[]
   MqlRates rates[];
//--- contador de intentos
   int attempts=0;
//--- cantidad que se ha copiado ya
   int copied=0;
//--- hacemos 25 intentos de obtener la serie temporal para el símbolo necesario
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si no se ha podido copiar la cantidad suficiente de barras
   if(copied!=bars)
     {
      //--- formaremos la cadena del mensaje
      string comm=StringFormat("Para el símbolo %s se ha logrado obtener sólo %d barra
                               name,
                               copied,
                               bars
                               );
      //--- mostraremos el mensaje en un comentario en la ventana principal del gráfic
      Comment(comm);
      //--- mostramos el mensaje
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- estableceremos la visualización del símbolo
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
     }
//--- inicializaremos los búfers con valores vacíos
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- en cada tic copiamos los precios a los búfers
   for(int i=0;i<copied;i++)
     {
      //--- calcularemos el índice correspondiente para los búfers
      int buffer_index=total-copied+i;
      //--- escribimos los precios en los búfers
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| devuelve aleatoriamente el símbolo desde Market Watch            |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- número de símbolos mostrados en la ventana "Observación del mercado"
   int symbols=SymbolsTotal(true);
//--- posición del símbolo en la lista - un número aleatorio de 0 a symbols
   int number=MathRand()%symbols;
//--- devolvemos el nombre del símbolo sobre la posición indicada
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de las barras                               |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de las barras
   string comm="";
//--- bloque del cambio del color de las barras
   int number=MathRand(); // obtenemos un número aleatorio
//--- el divisor del número es igual al tamaño del array colors[]
   int size=ArraySize(colors);
//--- obtenemos el índice para seleccionar nuevo color como el remanente de la divisi
   int color_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- apuntamos el color
   comm=comm+"\r\n"+(string)colors[color_index];
//--- apuntamos el nombre del símbolo
   comm="\r\n"+symbol+comm;
//--- mostraremos la información en el gráfico a través del comentario
```

© 2000-2025, MetaQuotes Ltd.

```
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_LINE

El estilo DRAW_COLOR_LINE es la versión en color del estilo DRAW_LINE. Éste también traza una línea según los valores del búfer indicador. Pero este estilo, igual que todos los estilos de color en cuyo nombre figura **COLOR**, cuenta con un especial búfer indicador adicional que guarda el índice (número) del color desde un especial array de colores. De esta manera, se puede definir el color de cada sección de la línea indicando el índice del color que debe adquirir la línea en una barra en concreto.

Usted puede establecer el grosor, color y el estilo de la línea tanto con las directivas del compilador, como de forma dinámica, utilizando la función PlotIndexSetInteger(). La opción del cambio dinámico de las propiedades de la construcción gráfica permite crear indicadores "vivos", es decir, los que cambian su apariencia en función de la situación actual.

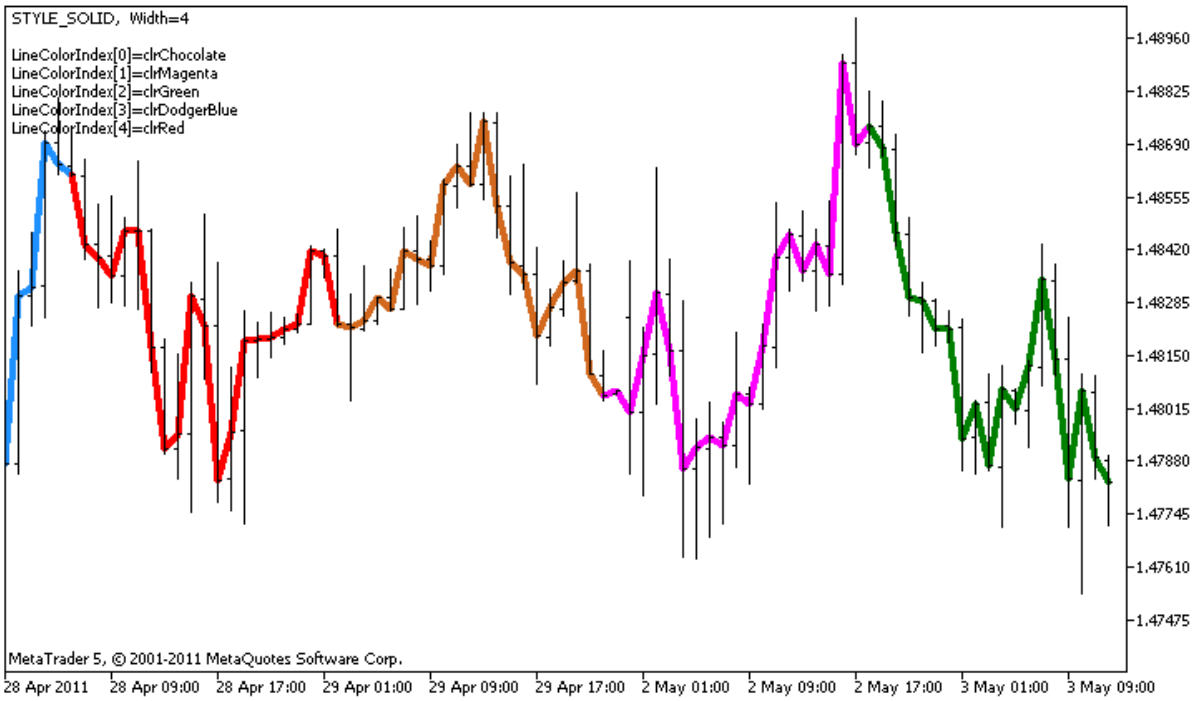
El número de búfers requeridos para construir DRAW_COLOR_LINE — 2:

- un búfer para almacenar los valores del indicador a base de los cuales se traza la línea;
- un búfer para almacenar el índice de color con el que se traza la línea en cada barra.

Se puede definir los colores con la directiva del compilador #property indicator_color1, separados por coma. El número de colores no puede superar 64.

```
//--- estableceremos 5 colores para colorear cada barra (se almacenan en un array espe
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (se puede
```

Aquí tenemos un ejemplo del indicador que traza una línea usando los precios de cierre de las barras Close. El grosor y el estilo de la línea se cambian de forma aleatoria cada N=5 tics.



Los colores para los segmentos de la línea también se cambian aleatoriamente en la función personalizada ChangeColors().

```
//+------------------------------------------------------------------+
//|  cambia el color de segmentos de la línea                        |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- número de colores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- definimos de forma aleatoria un color nuevo para cada índice de colores
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- obtendremos un número aleatorio
      int number=MathRand();
      //--- obtendremos un índice en el array col[] como el remanente de la división c
      int i=number%size;
      //--- estableceremos el color para cada índice como la propiedad PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  número del estilo gráfico
                          PLOT_LINE_COLOR,      //  identificador de la propiedad
                          plot_color_ind,       //  índice del color donde escribiremo
                          cols[i]);             //  nuevo color
      //--- apuntaremos los colores
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
```

En el ejemplo se muestra una particularidad propia de las versiones "de colores" de los indicadores - para cambiar el color de un segmento de la línea no hace falta cambiar los valores en el búfer ColorLineColors[] (que almacena los índices de colores). Basta con definir nuevos colores en un array especial. Esto permite cambiar rápidamente el color para toda la construcción gráfica, modificando únicamente un pequeño array de colores a través de la función PlotIndexSetInteger().

Fíjense que inicialmente para la construcción gráfica **plot1** con el estilo DRAW_COLOR_LINE las propiedades se establecen mediante la directiva del compilador #property, y luego en la función OnCalculate() estas tres propiedades se establecen de forma aleatoria.

Los parámetros N y Length (longitud de los segmentos coloreados de la barra) están sacados a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_LINE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Indicador para demostrar DRAW_COLOR_LINE"
#property description "Dibuja con segmentos de colores de 20 barras la línea a base de
#property description "El color, grosor y el estilo de segmentos de la línea se cambia
#property description "cada N tics"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
//--- estableceremos 5 colores para colorear cada barra (se almacenan en un array espe
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (se puede
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parámetros input
input int      N=5;           // número de tics a cambiar
input int      Length=20;     // longitud de cada sección de color en barras
int            line_colors=5; // número de colores definidos es igual a 5 - vea más ar
//--- búfer para dibujar
double         ColorLineBuffer[];
//--- búfer para almacenar el color del trazado de la línea en cada barra
double         ColorLineColors[];

//--- el array para almacenar colores contiene 7 elementos
color colors[]={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGolde
//--- array para almacenar estilos de trazado de la línea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array con el búfer indicador
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- inicialización del generador de números pseudoaleatorios
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número crítico de tics,
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- cambiamos los colores con los que se dibujan los segmentos de colores de l
      ChangeColors(colors,5);
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- bloque para calcular los valores del indicador
   for(int i=0;i<rates_total;i++)
     {
      //--- apuntamos el valor del indicador en el búfer
      ColorLineBuffer[i]=close[i];
      //--- ahora de forma aleatoria definimos un índice de color para esta barra
      int color_index=i%(5*Length);
      color_index=color_index/Length;
      //--- para esta barra la línea va a dibujarse con el color que se guarda bajo el
      ColorLineColors[i]=color_index;
     }

//--- volveremos el valor prev_calculated para la siguiente llamada de la función
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  cambia el color de segmentos de la línea                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- número de colores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- definimos de forma aleatoria un color nuevo para cada índice de colores
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- obtendremos un número aleatorio
```

© 2000-2025, MetaQuotes Ltd.

```
      int number=MathRand();
      //--- obtendremos un índice en el array col[] como el remanente de la división c
      int i=number%size;
      //--- estableceremos el color para cada índice como la propiedad PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  número del estilo gráfico
                          PLOT_LINE_COLOR,      //  identificador de la propiedad
                          plot_color_ind,       //  índice del color donde escribiremo
                          cols[i]);             //  nuevo color
      //--- apuntaremos los colores
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de la línea trazada en el indicador         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de la línea
   string comm="";
//--- bloque de cambio del grosor de la línea
   int number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5; // el grosor puede ser de 0 a 4
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+" Width="+IntegerToString(width);

//--- bloque de cambio del estilo de la línea
   number=MathRand();
//--- el divisor del número es igual al tamaño del array styles
   int size=ArraySize(styles);
//--- obtenemos el índice para seleccionar nuevo estilo como el remanente de la divisi
   int style_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- apuntaremos el estilo de la línea
   comm=EnumToString(styles[style_index])+", "+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_SECTION

El estilo DRAW_COLOR_SECTION es la versión de colores del estilo DRAW_SECTION, pero a diferencia del último, permite colorear cada segmento con su propio color. El estilo DRAW_COLOR_SECTION, igual que todos los estilos de color en cuyo nombre figura **COLOR**, cuenta con un especial búfer indicador adicional que guarda el índice (número) del color desde un especial array de colores. De esta manera, se puede definir el color de cada segmento indicando un índice de color para la barra en la que cae el fin del segmento.

El grosor, color y el estilo de segmentos se puede establecer de la misma manera como para el estilo DRAW_SECTION - con las directivas del compilador, o bien dinámicamente, utilizando la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite "vivificar" los indicadores, para que cambien su apariencia en función de la situación actual.

Los segmentos se trazan desde un valor no vacío hasta otro valor no vacío del búfer indicador, ignorando los valores vacíos. Para indicar qué valor se debe considerar "vacío", establezca este valor en la propiedad PLOT_EMPTY_VALUE. Por ejemplo, si un indicador debe dibujarse con segmentos sobre los valores no nulos, entonces hay que establecer el valor nulo como vacío:

```
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetDouble(índice_ de_construcción_DRAW_COLOR_SECTION,PLOT_EMPTY_VALUE,0);
```

Rellene siempre todos los elementos del búfer indicador con valores de forma explícita, estableciendo el valor vacío para los elementos que no van a dibujarse.

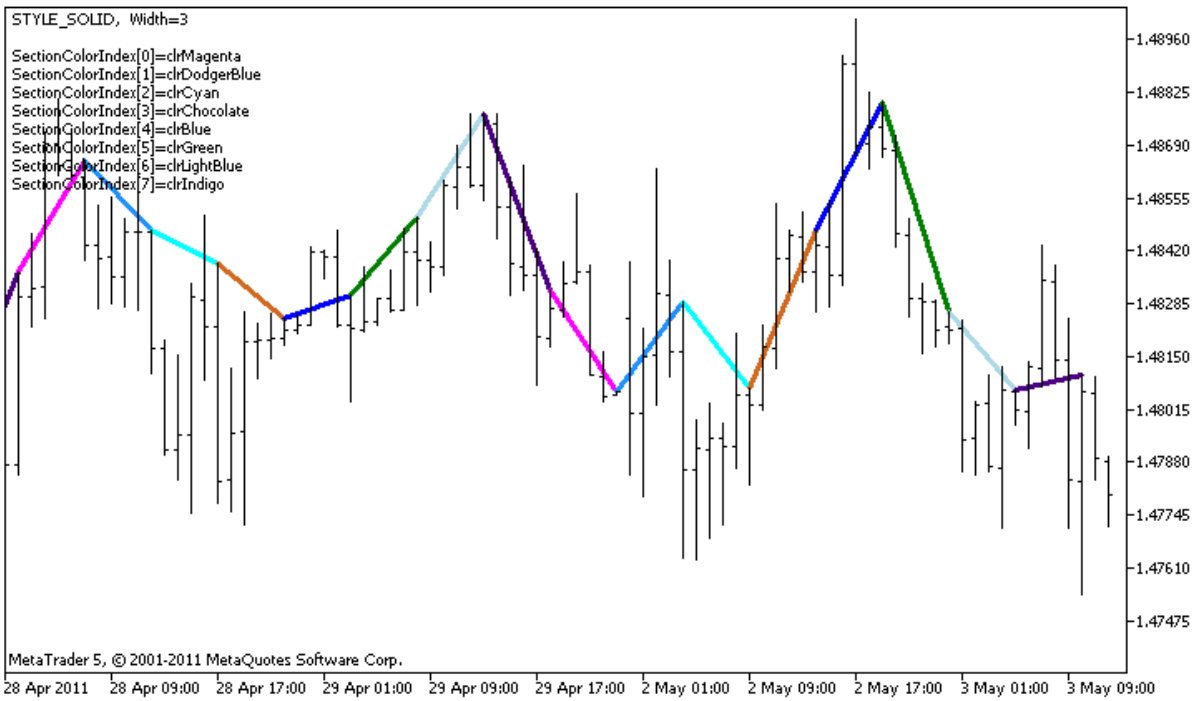
El número de búfers requeridos para construir DRAW_COLOR_SECTION — 2:

- un búfer para almacenar los valores del indicador a base de los cuales se traza la línea;
- un búfer para almacenar el índice de color con el que se colorea el segmento (tiene sentido establecer sólo para los valores no vacíos).

Se puede definir los colores con la directiva del compilador #property indicator_color1, separados por coma. El número de colores no puede superar 64.

Aquí tenemos un ejemplo del indicador que dibuja segmentos de diferentes colores de 5 barras de longitud a base de los precios High. El color, grosor y el estilo de segmentos se cambian de forma aleatoria cada **N** tics.

© 2000-2025, MetaQuotes Ltd.



Fíjense, inicialmente para la construcción gráfica **plot1** con el estilo DRAW_COLOR_SECTION se establecen 8 colores mediante la directiva del compilador #property. Luego en la función OnCalculate() los colores se establecen de forma aleatoria desde el array de colores colors[].

El parámetro N está pasado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                            DRAW_COLOR_SECTION.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_COLOR_SECTION"
#property description "Dibuja con segmentos de colores de la longitud que corresponde
#property description "El color, grosor y el estilo del segmento se cambia de forma al
#property description "dentro de cada N tics"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorSection
#property indicator_label1  "ColorSection"
#property indicator_type1   DRAW_COLOR_SECTION
//--- estableceremos 8 colores para colorear los segmentos (se guardan en un array esp
#property indicator_color1  clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,c
#property indicator_style1  STYLE_SOLID
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_width1  1
//--- parámetros input
input int      N=5;                      // número de tics a cambiar
input int      bars_in_section=5;        // longitud de segmentos en barras
//--- una variable auxiliar para calcular los extremos de segmentos
int            divider;
int            color_sections;
//--- búfer para dibujar
double         ColorSectionBuffer[];
//--- búfer para almacenar el color del trazado de la línea en cada barra
double         ColorSectionColors[];
//--- el array para almacenar colores tiene 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- array para almacenar estilos de trazado de la línea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- número de colores para colorear los segmentos
   color_sections=8;   //  ver comentario para la propiedad #property indicator_color1
//--- comprobaremos el parámetro del indicador
   if(bars_in_section<=0)
     {
      PrintFormat("Segmento tiene una longitud inválida=%d",bars_in_section);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=color_sections*bars_in_section;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número crítico de tics,
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- cambiamos colores con los que se dibujan segmentos
      ChangeColors(colors,color_sections);
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- número de la barra a partir del cual empezaremos a calcular los valores del indi
   int start=0;
//--- si el indicador ha sido calculado antes, estableceremos start para la barra ante
   if(prev_calculated>0) start=prev_calculated-1;
//--- aquí están todos los cálculos de los valores del indicador
   for(int i=start;i<rates_total;i++)
     {
      //--- si el número de la barra se divide sin remanente por la longitud_del_segme
      if(i%bars_in_section==0)
        {
         //--- estableceremos el extremo del segmento en el precio High de esta barra
         ColorSectionBuffer[i]=high[i];
         //--- remanente de la división del número de la barra por longitud_del_segmen
         int rest=i%divider;
         //obtendremos el número del color =  de 0 a número_de_colores-1
         int color_indext=rest/bars_in_section;
         ColorSectionColors[i]=color_indext;
        }
      //--- si la remanente de la división es igual a bars,
      else
        {
         //--- si no encaja nada, omitimos esta barra - ponemos el valor 0
         ColorSectionBuffer[i]=0;
        }
     }
//--- volveremos el valor prev_calculated para la siguiente llamada de la función
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  cambia el color de segmentos de la línea                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- número de colores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- definimos de forma aleatoria un color nuevo para cada índice de colores
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- obtendremos un número aleatorio
      int number=MathRand();
      //--- obtendremos un índice en el array col[] como el remanente de la división c
      int i=number%size;
      //--- estableceremos el color para cada índice como la propiedad PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  número del estilo gráfico
                          PLOT_LINE_COLOR,      //  identificador de la propiedad
                          plot_color_ind,       //  índice del color donde escribiremo
                          cols[i]);             //  nuevo color
      //--- apuntaremos los colores
      comm=comm+StringFormat("SectionColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de la línea trazada en el indicador         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de la línea
   string comm="";
//--- bloque de cambio del grosor de la línea
   int number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5; // el grosor puede ser de 0 a 4
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+" Width="+IntegerToString(width);

//--- bloque de cambio del estilo de la línea
   number=MathRand();
//--- el divisor del número es igual al tamaño del array styles
   int size=ArraySize(styles);
//--- obtenemos el índice para seleccionar nuevo estilo como el remanente de la divisi
   int style_index=number%size;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- apuntaremos el estilo de la línea
   comm=EnumToString(styles[style_index])+", "+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM

El estilo DRAW_COLOR_HISTOGRAM dibuja un histograma de columnas de diferentes colores desde cero hasta el valor especificado. Los valores se cogen desde el búfer indicador. Cada columna puede tener su propio color elegido de un conjunto previamente definido.

El grosor, color y el estilo del histograma se puede establecer de la misma manera como para el estilo DRAW_HISTOGRAM - con las directivas del compilador, o bien dinámicamente, utilizando la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite cambiar la apariencia del histograma en función de la situación actual.

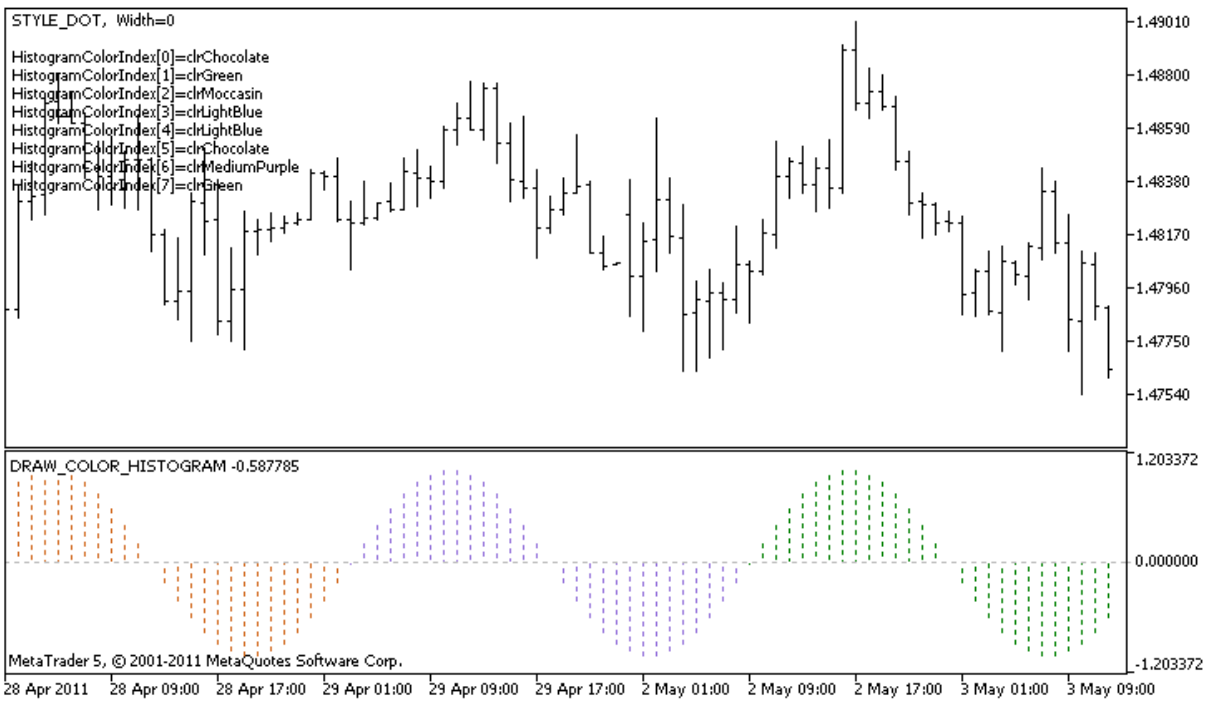
Ya que en cada barra se dibuja una columna desde el nivel cero, es mejor utilizar DRAW_COLOR_HISTOGRAM para mostrar en otra subventana del gráfico. Las más de las veces este tipo de construcción gráfica se utiliza para crear los indicadores del tipo oscilatorio, por ejemplo: Awesome Oscillator o Market Facilitation Index. Para los valores vacíos que no se muestran, será suficiente indicarlos valores nulos.

El número de búfers requeridos para construir DRAW_COLOR_HISTOGRAM — 2:

- un búfer para almacenar el valor no nulo del segmento vertical en cada barra, el segundo extremos del segmento siempre se encuentra en la línea cero del indicador;
- un búfer para almacenar el índice de color con el que se colorea el segmento (tiene sentido establecer sólo para los valores no vacíos).

Se puede definir los colores con la directiva del compilador #property indicator_color1, separados con coma. El número de colores no puede superar 64.

Aquí tenemos un ejemplo del indicador que dibuja una sinusoide de color especificado basándose en la función MathSin(). El color, grosor y el estilo de **todas** las columnas se cambian de forma aleatoria cada **N** tics. El parámetro bars determina el período de la sinusoide, eso quiere decir que dentro de una cantidad de barras especificada la sinusoide va a repetir su ciclo.



© 2000-2025, MetaQuotes Ltd.

Fíjense, inicialmente para la construcción gráfica **plot1** con el estilo DRAW_COLOR_HISTOGRAM se establecen 5 colores mediante la directiva del compilador #property indicator_color1, y luego en la función OnCalculate() estos colores se eligen aleatoriamente de 14 colores que se guardan en el array colors[]. El parámetro N está pasado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_COLOR_HISTOGRAM"
#property description "Dibuja la sinusoide como un histograma en otra ventana"
#property description "El color y el grosor de las columnas se cambian de forma aleato
#property description "dentro de cada N tics"
#property description "El parámetro bars establece el número de barras para la repetic

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- parámetros input
input int      bars=30;          // período de la sinusoide en barras
input int      N=5;              // número de tics para el cambio del histograma
//--- plot Color_Histogram
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- estableceremos 8 colores para colorear los segmentos (se guardan en un array esp
#property indicator_color1  clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrMe
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- búfer de valores
double         Color_HistogramBuffer[];
//--- búfer para los índices de colores
double         Color_HistogramColors[];
//--- factor para obtener el ángulo 2Pi en radiánes al multiplicar por el parámetro ba
double         multiplier;
int            color_sections;
//--- el array para almacenar colores tiene 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- array para almacenar estilos de trazado de la línea
```

© 2000-2025, MetaQuotes Ltd.

```
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
//---- número de colores para colorear la sinusoide
   color_sections=8;   //  ver comentario para la propiedad #property indicator_color1
//--- calcularemos el multiplicador
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Establezca el valor bars=%d mayor que 1",bars);
      //--- finalización anticipada del trabajo del indicador
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número crítico de tics,
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- cambiamos los colores con los que se dibuja el histograma
      ChangeColors(colors,color_sections);
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- cálculos de los valores del indicador
   int start=0;
//--- si el cálculo ya ha sido realizado en el arranque anterior de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // fijaremos el inicio del cálculo a
//--- llenamos el búfer indicador con valores
   for(int i=start;i<rates_total;i++)
     {
      //--- valor
      Color_HistogramBuffer[i]=sin(i*multiplier);
      //--- color
      int color_index=i%(bars*color_sections);
      color_index/=bars;
      Color_HistogramColors[i]=color_index;
     }
//--- volveremos el valor prev_calculated para la siguiente llamada de la función
   return(rates_total);
  }
  //+------------------------------------------------------------------+
//|  cambia el color de segmentos de la línea                          |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- número de colores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- definimos de forma aleatoria un color nuevo para cada índice de colores
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- obtendremos un número aleatorio
      int number=MathRand();
      //--- obtendremos un índice en el array col[] como el remanente de la división c
      int i=number%size;
      //--- estableceremos el color para cada índice como la propiedad PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  número del estilo gráfico
                          PLOT_LINE_COLOR,      //  identificador de la propiedad
                          plot_color_ind,       //  índice del color donde escribiremo
                          cols[i]);             //  nuevo color
      //--- apuntaremos los colores
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de la línea trazada en el indicador         |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de la línea
   string comm="";
//--- bloque de cambio del grosor de la línea
   int number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5; // el grosor puede ser de 0 a 4
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+" Width="+IntegerToString(width);

//--- bloque de cambio del estilo de la línea
   number=MathRand();
//--- el divisor del número es igual al tamaño del array styles
   int size=ArraySize(styles);
//--- obtenemos el índice para seleccionar nuevo estilo como el remanente de la divisi
   int style_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- apuntaremos el estilo de la línea
   comm=EnumToString(styles[style_index])+", "+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM2

El estilo DRAW_COLOR_HISTOGRAM2 dibuja un histograma de color especificado – segmentos verticales, usando valores de dos búfers indicadores. Pero a diferencia del estilo DRAW_HISTOGRAM2 unicolor, en este estilo cada columna del histograma puede obtener su propio color desde el conjunto predefinido. Los valores de los extremos de segmentos se cogen del búfer indicador.

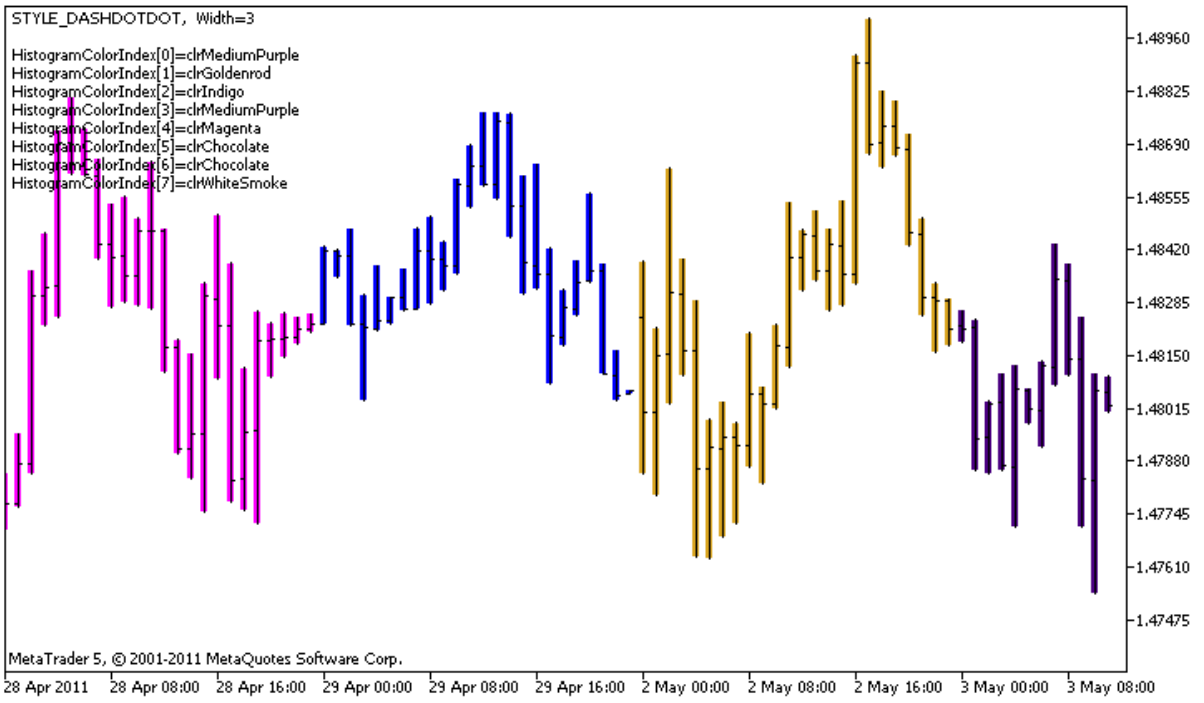
El grosor, colores y el estilo del histograma se puede establecer de la misma manera como para el estilo DRAW_HISTOGRAM2 – con las directivas del compilador, o bien dinámicamente, utilizando la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite cambiar la apariencia del histograma en función de la situación actual.

El estilo DRAW_COLOR_HISTOGRAM2 se puede utilizar tanto en una subventana separada del gráfico, como en la ventana principal. Los valores vacíos no se dibujan, todos los valores hay que establecer en los búfers indicadores de forma explícita. Los búfers no se inicializan con valores vacíos.

El número de búfers requeridos para construir DRAW_COLOR_HISTOGRAM2 — 3:

- dos búfers para guardar el extremo superior e inferior del segmento vertical en cada barra;
- un búfer para almacenar el índice de color con el que se colorea el segmento (tiene sentido establecer sólo para los valores no vacíos).

Aquí tenemos un ejemplo del indicador que dibuja con un color especificado un histograma entre los precios High y Low. Para cada día de la semana las líneas del histograma tendrán su color. El color de cada día, grosor y el estilo del histograma se cambian de forma aleatoria cada **N** tics.



Fíjense, inicialmente para la construcción gráfica **plot1** con el estilo DRAW_COLOR_HISTOGRAM2 se establecen 5 colores mediante la directiva del compilador #property indicator_color1, y luego en la función OnCalculate() estos colores se eligen aleatoriamente de 14 colores que se guardan en el array colors[].

© 2000-2025, MetaQuotes Ltd.

El parámetro N está pasado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                        DRAW_COLOR_HISTOGRAM2.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_COLOR_HISTOGRAM2"
#property description "Dibuja en cada barra un segmento entre Open y Close"
#property description "El color, grosor y el estilo se cambian de forma aleatoria"
#property description "dentro de cada N tics"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plot ColorHistogram_2
#property indicator_label1  "ColorHistogram_2"
#property indicator_type1   DRAW_COLOR_HISTOGRAM2
//--- estableceremos 5 colores para colorear el histograma por días de la semana (se a
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- parámetro input
input int      N=5;              // número de tics para el cambio del histograma
int            color_sections;
//--- búfers de valores
double         ColorHistogram_2Buffer1[];
double         ColorHistogram_2Buffer2[];
//--- búfer para los índices del color
double         ColorHistogram_2Colors[];
//--- el array para almacenar colores tiene 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- array para almacenar estilos de trazado de la línea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
//--- estableceremos el valor vacío
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- número de colores para colorear la sinusoide
   color_sections=8;   //  ver comentario para la propiedad #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número crítico de tics,
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- cambiamos los colores con los que se dibuja el histograma
      ChangeColors(colors,color_sections);
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- cálculos de los valores del indicador
   int start=0;
//--- para obtener el día de la semana por la hora de apertura de cada barra
   MqlDateTime dt;
//--- si el cálculo ya ha sido realizado en el arranque anterior de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // fijaremos el inicio del cálculo a
//--- llenamos el búfer indicador con valores
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- valor
      ColorHistogram_2Buffer1[i]=high[i];
      ColorHistogram_2Buffer2[i]=low[i];
      //--- establecemos el índice de color según el día de la semana
      int day=dt.day_of_week;
      ColorHistogram_2Colors[i]=day;
     }
//--- volveremos el valor prev_calculated para la siguiente llamada de la función
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  cambia el color de segmentos de la línea                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- número de colores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- definimos de forma aleatoria un color nuevo para cada índice de colores
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- obtendremos un número aleatorio
      int number=MathRand();
      //--- obtendremos un índice en el array col[] como el remanente de la división c
      int i=number%size;
      //--- estableceremos el color para cada índice como la propiedad PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  número del estilo gráfico
                          PLOT_LINE_COLOR,      //  identificador de la propiedad
                          plot_color_ind,       //  índice del color donde escribiremo
                          cols[i]);             //  nuevo color
      //--- apuntaremos los colores
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de la línea trazada en el indicador         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de la línea
   string comm="";
//--- bloque de cambio del grosor de la línea
   int number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5; // el grosor puede ser de 0 a 4
```

© 2000-2025, MetaQuotes Ltd.

```
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+" Width="+IntegerToString(width);

//--- bloque de cambio del estilo de la línea
   number=MathRand();
//--- el divisor del número es igual al tamaño del array styles
   int size=ArraySize(styles);
//--- obtenemos el índice para seleccionar nuevo estilo como el remanente de la divisi
   int style_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- apuntaremos el estilo de la línea
   comm=EnumToString(styles[style_index])+", "+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_ARROW

El estilo DRAW_COLOR_ARROW dibuja en el gráfico las flechas de color (símbolos del conjunto Wingdings) basándose en el valor del búfer indicador. A diferencia del estilo DRAW_ARROW, este estilo permite establecer para cada símbolo su color desde un conjunto predefinido de colores especificados por medio de la propiedad indicator_color1.

El grosor y el color de los símbolo se puede establecer de la misma manera como para el estilo DRAW_ARROW - con las directivas del compilador, o bien dinámicamente, utilizando la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite cambiar la apariencia del indicador en función de la situación actual.

El código del símbolo a mostrar en el gráfico se establece a través de la propiedad PLOT_ARROW.

```
//--- estableceremos el código del símbolo desde el fuente Wingdings para dibujar en
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

Por defecto, el valor de PLOT_ARROW=159 (un círculo).

Cada flecha prácticamente es un símbolo que tiene su alto y su punto de enlace, y puede cubrir alguna información importante en el gráfico (por ejemplo, el precio del cierre en la barra). Por eso se puede indicar adicionalmente el desplazamiento vertical en píxeles, que no depende de la escala del gráfico. Las flechas se desplazarán visualmente por la línea vertical a esta especificada cantidad de píxeles, aunque los valores del indicador se quedarán los mismos:

```
//--- estableceremos el desplazamiento de flechas por la línea vertical en píxeles
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

Un valor negativo de PLOT_ARROW_SHIFT significa el desplazamiento de las flechas hacia arriba, un valor positivo significa el desplazamiento de la flecha hacia abajo.

El estilo DRAW_COLOR_ARROW se puede utilizar tanto en una subventana separada del gráfico, como en la ventana principal. Los valores vacíos no se dibujan y no se muestran en la "Ventana de datos", todos los valores hay que establecer en los búfers indicadores de forma explícita. Los búfers no se inicializan con valores vacíos.

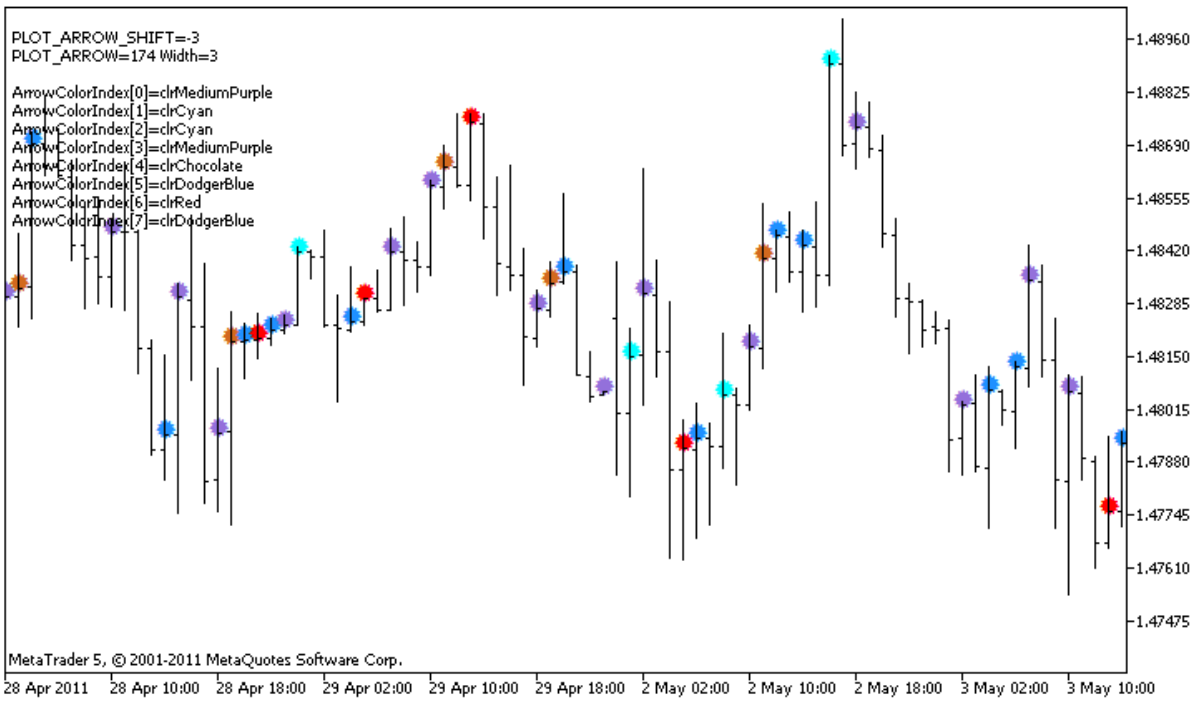
```
//--- estableceremos el valor vacío
   PlotIndexSetDouble(índice_ de_construcción_DRAW_COLOR_ARROW,PLOT_EMPTY_VALUE,0);
```

El número de búfers requeridos para construir DRAW_COLOR_ARROW — 2:

- un búfer para almacenar los valores del precio a base del cual se dibuja el símbolo (más el desplazamiento en píxeles que se fila por la propiedad PLOT_ARROW_SHIFT);
- un búfer para almacenar el índice de color con el que se colorea la flecha (tiene sentido establecer sólo para los valores no vacíos).

Aquí tenemos un ejemplo del indicador que dibuja las flechas en cada barra cuyo precio de cierre Close es más alto que el precio de cierre de la barra anterior. El grosor, desplazamiento y el código del símbolo de **todas** las flechas se cambian de forma aleatoria cada **N** tics. El precio del símbolo depende del número de la barra en la que está dibujado.

© 2000-2025, MetaQuotes Ltd.



En el ejemplo, inicialmente para la construcción gráfica **plot1** con el estilo DRAW_COLOR_ARROW las propiedades del color y el tamaño se establecen mediante la directiva del compilador #property, y luego en la función OnCalculate() las propiedades se cambian de forma aleatoria. El parámetro N está pasado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

Fíjense, inicialmente se establecen 8 colores mediante la directiva del compilador #property, y luego en la función OnCalculate() el color se elige aleatoriamente de 14 colores que se guardan en el array colors[].

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_ARROW.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_COLOR_ARROW"
#property description "Dibuja en el gráfico las flechas de diferentes colores determin
#property description "El color, tamaño, desplazamiento y el código del símbolo de la
#property description " de forma aleatoria cada N tics"
#property description "El parámetro code establece el valor base: código=159 (círculo)

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorArrow
#property indicator_label1  "ColorArrow"
```

```
#property indicator_type1   DRAW_COLOR_ARROW
//--- estableceremos 8 colores para colorear el histograma por días de la semana (se a
#property indicator_color1  clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagent
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- parámetros input
input int      N=5;         // número de tics para el cambio
input ushort   code=159;    // código del símbolo a dibujar en DRAW_ARROW
int            color_sections;
//--- búfer indicador para la construcción
double         ColorArrowBuffer[];
//--- búfer para guardar los índices del color
double         ColorArrowColors[];
//--- el array para almacenar colores tiene 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
//--- estableceremos el código del símbolo para dibujar en PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- estableceremos el desplazamiento de flechas por la línea vertical en píxeles
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- estableceremos un 0 como valor vacío
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- número de colores para colorear la sinusoide
   color_sections=8;   //  ver comentario para la propiedad #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del color, tamaño, desplazamiento y código de l
   ticks++;
//--- si tenemos acumulado un número crítico de tics,
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de las flechas
      ChangeLineAppearance();
      //--- cambiamos los colores con los que se dibuja el histograma
      ChangeColors(colors,color_sections);
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- bloque para calcular los valores del indicador
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- ciclo del cálculo
   for(int i=1;i<rates_total;i++)
     {
      //--- si el precio actual Close es más alto que el anterior, colocamos la flecha
      if(close[i]>close[i-1])
         ColorArrowBuffer[i]=close[i];
      //--- en caso contrario, mostramos el valor cero
      else
         ColorArrowBuffer[i]=0;
      //--- color de la flecha
      int index=i%color_sections;
      ColorArrowColors[i]=index;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  cambia el color de segmentos de la línea                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- número de colores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- definimos de forma aleatoria un color nuevo para cada índice de colores
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- obtendremos un número aleatorio
       int number=MathRand();
       //--- obtendremos un índice en el array col[] como el remanente de la división c
       int i=number%size;
       //--- estableceremos el color para cada índice como la propiedad PLOT_LINE_COLOR
       PlotIndexSetInteger(0,                    //  número del estilo gráfico
                           PLOT_LINE_COLOR,      //  identificador de la propiedad
                           plot_color_ind,       //  índice del color donde escribiremo
                           cols[i]);             //  nuevo color
       //--- apuntaremos los colores
       comm=comm+StringFormat("ArrowColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStrin
       ChartSetString(0,CHART_COMMENT,comm);
      }
//---
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de la línea trazada en el indicador         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de la línea
   string comm="";
//--- bloque de cambio del grosor de la línea
   int number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5; // el grosor puede ser de 0 a 4
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+" Width="+IntegerToString(width);

//--- bloque del cambio del código de la flecha (PLOT_ARROW)
   number=MathRand();
//--- obtendremos el remanente de la división de números enteros para calcular nuevo c
   int code_add=number%20;
//--- estableceremos el nuevo código del símbolo como la suma code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- apuntaremos el código del símbolo PLOT_ARROW
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- bloque del cambio del desplazamiento de flechas por la línea vertical en píxeles
   number=MathRand();
//--- obtenemos el desplazamiento como el remanente de la división de números enteros
   int shift=20-number%41;
//--- estableceremos nuevo desplazamiento
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- apuntaremos el desplazamiento PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_ZIGZAG

El estilo DRAW_COLOR_ZIGZAG dibuja segmentos de diferentes colores, usando valores de dos búfers indicadores. Este estilo es la versión de colores del estilo DRAW_ZIGZAG. Es decir, permite fijar para cada segmento su propio color desde un conjunto de colores predefinido previamente. Los segmentos se trazan desde un valor en el primer búfer indicador hasta un valor en el segundo. Ninguno de los dos búferes puede contener sólo valores vacíos. Si es así, no se dibuja nada.

El grosor, color y el estilo de la línea se puede establecer de la misma manera como para el estilo DRAW_ZIGZAG - con las directivas del compilador, o bien dinámicamente, utilizando la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite "vivificar" los indicadores, para que cambien su apariencia en función de la situación actual.

Los segmentos se trazan desde un valor no vacío de un búfer hasta otro valor no vacío de otro búfer indicador. Para indicar qué valor se debe considerar "vacío", establezca este valor en la propiedad PLOT_EMPTY_VALUE:

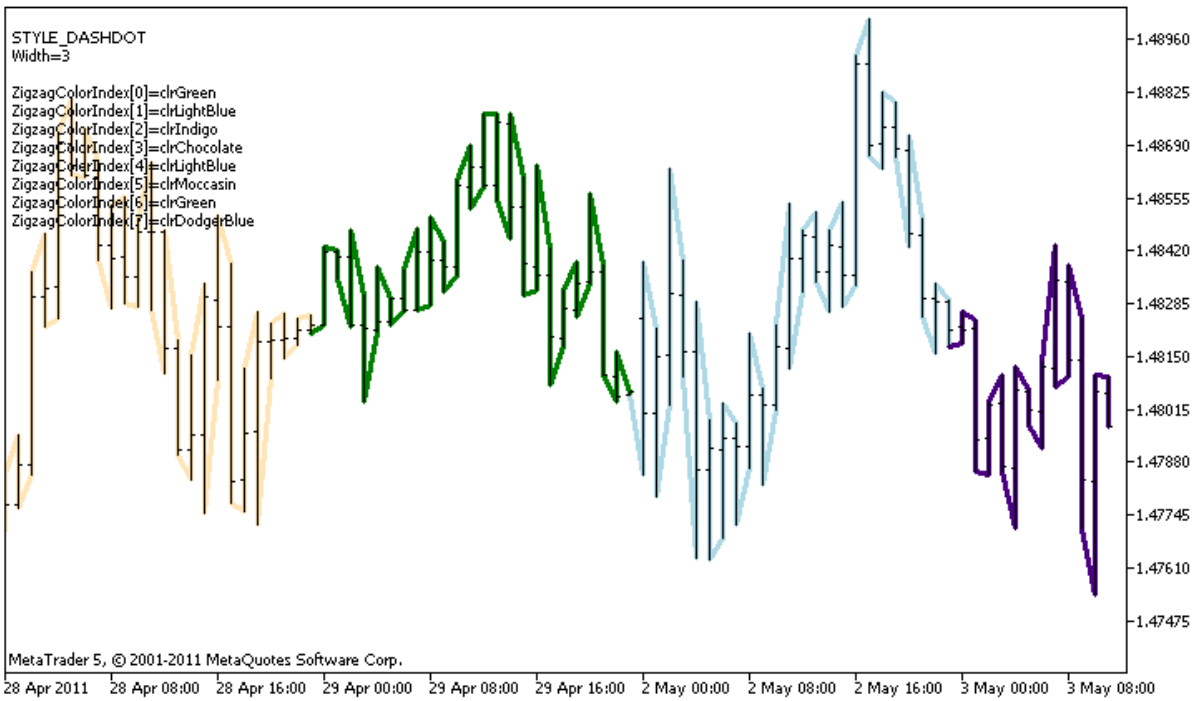
```
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetDouble(índice_ de_construcción_DRAW_COLOR_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Rellene siempre los búfers indicadores con valores de forma explícita, para las barras que se ignoran indique en el búfer un valor vacío.

El número de búfers requeridos para construir DRAW_COLOR_ZIGZAG — 3:

- dos búfers para almacenar los valores de los extremos de un segmento del zigzag;
- un búfer para almacenar el índice de color con el que se colorea el segmento (tiene sentido establecer sólo para los valores no vacíos).

Aquí tenemos un ejemplo del indicador que traza la sierra a base de los precios High y Low. El color, grosor y el estilo de la línea del zigzag se cambian de forma aleatoria cada N tics.



© 2000-2025, MetaQuotes Ltd.

Fíjense, inicialmente para la construcción gráfica **plot1** con el estilo DRAW_COLOR_ZIGZAG se establecen 8 colores mediante la directiva del compilador #property, y luego en la función OnCalculate() el color se elige aleatoriamente de 14 colores que se guardan en el array colors[].

El parámetro N está pasado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_ZIGZAG.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_COLOR_ZIGZAG"
#property description "Dibuja una línea quebrada con segmentos de colores, el color de
#property description "El color, grosor y el estilo de segmentos se cambia de forma al
#property description " dentro de cada N tics"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plot Color_Zigzag
#property indicator_label1  "Color_Zigzag"
#property indicator_type1   DRAW_COLOR_ZIGZAG
//--- estableceremos 8 colores para colorear los segmentos (se guardan en un array esp
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parámetro input
input int      N=5;              // número de tics a cambiar
int            color_sections;
//--- búfers de valores de los extremos de segmentos
double         Color_ZigzagBuffer1[];
double         Color_ZigzagBuffer2[];
//--- búfer de los índices de color para los extremos de los segmentos
double         Color_ZigzagColors[];
//--- el array para almacenar colores tiene 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- array para almacenar estilos de trazado de la línea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
//---- número de colores para colorear el zigzag
   color_sections=8;   //  ver comentario para la propiedad #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la línea
   ticks++;
//--- si tenemos acumulado un número suficiente de tics
   if(ticks>=N)
     {
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- cambiamos colores con los que se dibujan segmentos
      ChangeColors(colors,color_sections);
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }

//--- necesitaremos una estructura de tiempo para obtener el día de la semana de cada
   MqlDateTime dt;

//--- posición de inicio del cálculo
   int start=0;
//--- si el indicador se calcula en el tic anterior, empezamos el cálculo a partir del
   if(prev_calculated!=0) start=prev_calculated-1;
//--- ciclo de cálculos
   for(int i=start;i<rates_total;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      //--- apuntaremos en la estructura la hora de apertura de la barra
      TimeToStruct(time[i],dt);

      //--- si el número de la barra es par
      if(i%2==0)
        {
         //---  escribimos en el 1-r búfer High, en el 2-do Low
         Color_ZigzagBuffer1[i]=high[i];
         Color_ZigzagBuffer2[i]=low[i];
         //--- color del segmento
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
      //--- el número de la barra es impar
      else
        {
         //--- llenamos la barra en sentido inverso
         Color_ZigzagBuffer1[i]=low[i];
         Color_ZigzagBuffer2[i]=high[i];
         //--- color del segmento
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  cambia el color de segmentos del zigzag
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- número de colores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- definimos de forma aleatoria un color nuevo para cada índice de colores
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- obtendremos un número aleatorio
      int number=MathRand();
      //--- obtendremos un índice en el array col[] como el remanente de la división c
      int i=number%size;
      //--- estableceremos el color para cada índice como la propiedad PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  número del estilo gráfico
                          PLOT_LINE_COLOR,      //  identificador de la propiedad
                          plot_color_ind,       //  índice del color donde escribiremo
                          cols[i]);             //  nuevo color
      //--- apuntaremos los colores
```

© 2000-2025, MetaQuotes Ltd.

```
      comm=comm+StringFormat("ZigzagColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de segmentos en el zigzag                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de Color_ZigZag
   string comm="";
//--- bloque de cambio del grosor de la línea
   int number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5;   // el grosor puede ser de 0 a 4
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- bloque de cambio del estilo de la línea
   number=MathRand();
//--- el divisor del número es igual al tamaño del array styles
   int size=ArraySize(styles);
//--- obtenemos el índice para seleccionar nuevo estilo como el remanente de la divisi
   int style_index=number%size;
//--- estableceremos el color como la propiedad PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- apuntaremos el estilo de la línea
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_BARS

El estilo DRAW_COLOR_BARS dibuja las barras basándose en los valores de cuatro búfers indicadores que contienen los precios Open, High, Low y Close. Este estilo es una versión más avanzada del estilo DRAW_BARS y permite establecer para cada barra su propio color desde un conjunto de colores predefinido previamente. Se utiliza para crear sus propios indicadores personalizados en forma de barras, también en otra subventana del gráfico y para otros instrumentos financieros.

El color de las barras se puede definir con las directivas del compilador, o dinámicamente a través de la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite "vivificar" los indicadores, para que cambien su apariencia en función de la situación actual.

El indicador se dibuja sólo para las barras que tienen establecidos los valores no vacíos de **todos** los cuatro búfers indicadores. Para indicar qué valor se debe considerar "vacío", establezca este valor en la propiedad PLOT_EMPTY_VALUE:

```
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetDouble(índice_ de_construcción_DRAW_COLOR_BARS,PLOT_EMPTY_VALUE,0);
```

Rellene siempre los búfers indicadores con valores de forma explícita, para las barras que se ignoran indique en el búfer un valor vacío.

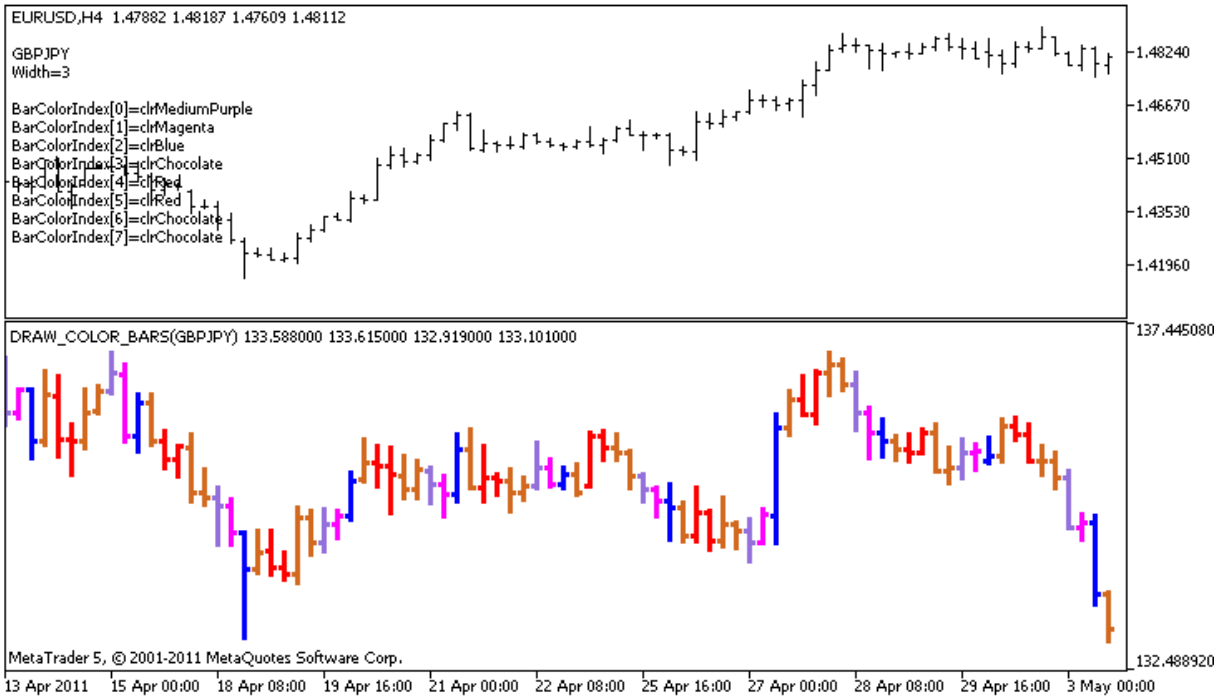
El número de búfers requeridos para construir DRAW_COLOR_BARS — 5:

- cuatro búfers para almacenar los valores Open, High, Low y Close;
- un búfer para almacenar el índice de color con el que se dibuja la barra (tiene sentido establecerlo sólo para las barras a dibujar).

Todos los búfers que se utilizan para la construcción deben ir en serie uno detrás del otro en orden establecido: Open, High, Low, Close y el búfer de color. Ninguno de los búfers de precios puede contener sólo los valores vacíos, porque en este caso no se dibuja nada.

Aquí tenemos un ejemplo del indicador que dibuja en una ventana separada las barras para el instrumento financiero especificado. El color de las barras se cambia aleatoriamente cada **N** tics. El parámetro N está sacado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

© 2000-2025, MetaQuotes Ltd.



Fíjense, inicialmente para la construcción gráfica **plot1** con el estilo DRAW_COLOR_BARS se establecen 8 colores mediante la directiva del compilador #property, y luego en la función OnCalculate() el color se elige aleatoriamente de 14 colores que se guardan en el array colors[].

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_BARS.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_COLOR_BARS"
#property description "Dibuja en la ventana separada las barras de diferentes colores
#property description "El color y el grosor de las barras, igual que el estilo, se car
#property description "cada N tics"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorBars
#property indicator_label1  "ColorBars"
#property indicator_type1   DRAW_COLOR_BARS
//--- estableceremos 8 colores para colorear las barras (se guardan en un array especi
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parámetros input
input int      N=5;              // número de tics para el cambio de apariencia
```

© 2000-2025, MetaQuotes Ltd.

```
input int      bars=500;         // número de barras a mostrar
input bool     messages=false;   // mostrar mensajes en el log "Asesores Expertos"
//--- búfers indicadores
double         ColorBarsBuffer1[];
double         ColorBarsBuffer2[];
double         ColorBarsBuffer3[];
double         ColorBarsBuffer4[];
double         ColorBarsColors[];
//--- nombre del símbolo
string symbol;
int    bars_colors;
//--- el array para almacenar colores tiene 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorBarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorBarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorBarsBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorBarsColors,INDICATOR_COLOR_INDEX);
//---- número de colores para colorear las barras
   bars_colors=8;   //  ver comentario para la propiedad #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- contamos los tics para el cambio del estilo, color y grosor de la barra
```

© 2000-2025, MetaQuotes Ltd.

```
   ticks++;
//--- si tenemos acumulado un número suficiente de tics
   if(ticks>=N)
     {
      //--- escogeremos nuevo símbolo en la ventana "Observación del mercado"
      symbol=GetRandomSymbolName();
      //--- cambiamos las propiedades de la línea
      ChangeLineAppearance();
      //--- cambiamos los colores con los que se dibujan las barras
      ChangeColors(colors,bars_colors);
      int tries=0;
      //--- haremos 5 intentos de llenar el búfer con los precios desde symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
         //--- contador de llamadas a la función CopyFromSymbolToBuffers()
         tries++;
        }
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  Llenamos los búfers indicadores con los precios                 |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
  {
//--- vamos a copiar los precios Open, High, Low y Close al array rates[]
   MqlRates rates[];
//--- contador de intentos
   int attempts=0;
//--- cantidad que se ha copiado ya
   int copied=0;
//--- hacemos 25 intentos de obtener la serie temporal para el símbolo necesario
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si no se ha podido copiar la cantidad suficiente de barras
   if(copied!=bars)
     {
      //--- formaremos la cadena del mensaje
      string comm=StringFormat("Para el símbolo %s se ha logrado obtener sólo %d barra
                               name,
                               copied,
                               bars
```

© 2000-2025, MetaQuotes Ltd.

```
                                  );
      //--- mostraremos el mensaje en un comentario en la ventana principal del gráfic
      Comment(comm);
      //--- mostramos el mensaje
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- estableceremos la visualización del símbolo
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS("+name+")");
     }
//--- inicializaremos los búfers con valores vacíos
   ArrayInitialize(ColorBarsBuffer1,0.0);
   ArrayInitialize(ColorBarsBuffer2,0.0);
   ArrayInitialize(ColorBarsBuffer3,0.0);
   ArrayInitialize(ColorBarsBuffer4,0.0);

//--- copiamos los precios a los búfers
   for(int i=0;i<copied;i++)
     {
      //--- calcularemos el índice correspondiente para los búfers
      int buffer_index=total-copied+i;
      //--- escribimos los precios en los búfers
      ColorBarsBuffer1[buffer_index]=rates[i].open;
      ColorBarsBuffer2[buffer_index]=rates[i].high;
      ColorBarsBuffer3[buffer_index]=rates[i].low;
      ColorBarsBuffer4[buffer_index]=rates[i].close;
      //---
      ColorBarsColors[buffer_index]=i%bar_colors;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| devuelve aleatoriamente el símbolo desde Market Watch            |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- número de símbolos mostrados en la ventana "Observación del mercado"
   int symbols=SymbolsTotal(true);
//--- posición del símbolo en la lista - un número aleatorio de 0 a symbols
   int number=MathRand()%symbols;
//--- devolvemos el nombre del símbolo sobre la posición indicada
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//|  cambia el color de segmentos del zigzag
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- número de colores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- definimos de forma aleatoria un color nuevo para cada índice de colores
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- obtendremos un número aleatorio
      int number=MathRand();
      //--- obtendremos un índice en el array col[] como el remanente de la división c
      int i=number%size;
      //--- estableceremos el color para cada índice como la propiedad PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  número del estilo gráfico
                          PLOT_LINE_COLOR,      //  identificador de la propiedad
                          plot_color_ind,       //  índice del color donde escribiremo
                          cols[i]);             //  nuevo color
      //--- apuntaremos los colores
      comm=comm+StringFormat("BarColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de las barras                               |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de las barras
   string comm="";

//--- bloque del cambio del grosor de las barras
   int number=MathRand();
//--- obtenemos el grosor como el remanente de la división de números enteros
   int width=number%5;   // el grosor puede ser de 0 a 4
//--- estableceremos el color como la propiedad PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- apuntaremos el grosor de la línea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- apuntamos el nombre del símbolo
   comm="\r\n"+symbol+comm;

//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_CANDLES

El estilo DRAW_COLOR_CANDLES, igual que el DRAW_CANDLES, dibuja las velas japonesas basándose en los valores de cuatro búfers indicadores que contienen los precios Open, High, Low y Close. Pero además de eso, este estilo permite establecer un color para cada vela desde un conjunto predefinido. Para eso, en el estilo ha sido agregado un búfer especial de colores que guarda los índices de los colores para cada barra. Se utiliza para crear sus propios indicadores personalizados en forma de velas japonesas, también en otra subventana del gráfico y para otros instrumentos financieros.

Usted puede establecer el número de colores para colorear las velas a través de las directivas del compilador, o dinámicamente a través de la función PlotIndexSetInteger(). El cambio dinámico de las propiedades de la construcción gráfica permite "vivificar" los indicadores, para que cambien su apariencia en función de la situación actual.

El indicador se dibuja sólo para las barras que tienen establecidos los valores no vacíos en los cuatro búfers de precios. Para indicar qué valor se debe considerar "vacío", establezca este valor en la propiedad PLOT_EMPTY_VALUE:

```
//--- el valor 0 (valor vacío) no va a participar en el proceso de trazado
   PlotIndexSetDouble(índice_ de_construcción_DRAW_COLOR_CANDLES,PLOT_EMPTY_VALUE,0);
```

Rellene siempre los búfers indicadores con valores de forma explícita, para las barras que se ignoran indique en el búfer un valor vacío.

El número de búfers requeridos para construir DRAW_COLOR_CANDLES — 5:

- cuatro búfers para almacenar los valores Open, High, Low y Close;
- un búfer para almacenar el índice de color con el que se dibuja la vela (tiene sentido establecerlo sólo para las velas a dibujar).

Todos los búfers que se utilizan para la construcción deben ir en serie uno detrás del otro en orden establecido: Open, High, Low, Close y el búfer de color. Ninguno de los búfers de precios puede contener sólo los valores vacíos, porque en este caso no se dibuja nada.

Aquí tenemos un ejemplo del indicador que dibuja en una ventana separada las velas japonesas para el instrumento financiero especificado. El color de las velas se cambia aleatoriamente cada **N** tics. El parámetro N está sacado a los parámetros externos del indicador para que exista la posibilidad de establecerlo manualmente (pestaña "Parámetros" en la ventana de propiedades del indicador).

© 2000-2025, MetaQuotes Ltd.



Fíjense en que inicialmente para la construcción gráfica **plot1** el color se establece mediante la directiva del compilador #property, y luego en la funciónOnCalculate() se elige aleatoriamente un nuevo color desde la lista previamente preparada.

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_CANDLES.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indicador para demostrar DRAW_COLOR_CANDLES."
#property description "Dibuja en la ventana separada las velas para el símbolo selecci
#property description " "
#property description "El color y el grosor de las velas, igual que el estilo, se camb
#property description "de forma aleatoria cada N tics."

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
//--- estableceremos 8 colores para colorear las velas (se guardan en un array especia
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```
//--- parámetros input
input int      N=5;              // número de tics para el cambio de apariencia
input int      bars=500;         // número de velas a mostrar
input bool     messages=false;   // mostrar mensajes en el log "Asesores Expertos"
//--- búfers indicadores
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- nombre del símbolo
string symbol;
//--- el array para almacenar colores tiene 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- si bars es muy pequeño - finalizamos el trabajo antes de tiempo
   if(bars<50)
     {
      Comment("¡Por favor, indique el número más grande de barras! El trabajo del indi
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorCandlesBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorCandlesBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorCandlesBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorCandlesBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- valor vacío
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- nombre del símbolo para el que se dibujan las barras
   symbol=_Symbol;
//--- estableceremos la visualización del símbolo
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
//---- número de colores para colorear las velas
   candles_colors=8;     //  ver comentario para la propiedad #property indicator_colo
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- contamos los tics para el cambio del estilo y color
   ticks++;
//--- si tenemos acumulado un número suficiente de tics
   if(ticks>=N)
     {
      //--- escogeremos nuevo símbolo en la ventana "Observación del mercado"
      symbol=GetRandomSymbolName();
      //--- cambiamos la apariencia
      ChangeLineAppearance();
      //--- cambiamos los colores con los que se dibujan las barras
      ChangeColors(colors,candles_colors);

      int tries=0;
      //--- haremos 5 intentos de llenar el búfer plot1 con los precios desde symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
            ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
            && tries<5)
        {
         //--- contador de llamadas a la función CopyFromSymbolToBuffers()
         tries++;
        }
      //--- actualizamos el contador de tics pasándolo a cero
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  Llena la vela indicada                                          |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
```

© 2000-2025, MetaQuotes Ltd.

```
                              double &buff2[],
                              double &buff3[],
                              double &buff4[],
                              double &col_buffer[],
                              int    cndl_colors
                              )
  {
//--- vamos a copiar los precios Open, High, Low y Close al array rates[]
   MqlRates rates[];
//--- contador de intentos
   int attempts=0;
//--- cantidad que se ha copiado ya
   int copied=0;
//--- hacemos 25 intentos de obtener la serie temporal para el símbolo necesario
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si no se ha podido copiar la cantidad suficiente de barras
   if(copied!=bars)
     {
      //--- formaremos la cadena del mensaje
      string comm=StringFormat("Para el símbolo %s se ha logrado obtener sólo %d barra
                               name,
                               copied,
                               bars
                               );
      //--- mostraremos el mensaje en un comentario en la ventana principal del gráfic
      Comment(comm);
      //--- mostramos el mensaje
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- estableceremos la visualización del símbolo
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
     }
//--- inicializaremos los búfers con valores vacíos
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- en cada tic copiamos los precios a los búfers
   for(int i=0;i<copied;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- calcularemos el índice correspondiente para los búfers
      int buffer_index=total-copied+i;
      //--- escribimos los precios en los búfers
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
      //--- estableceremos el color de la vela
      int color_index=i%cndl_colors;
      col_buffer[buffer_index]=color_index;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| devuelve aleatoriamente el símbolo desde Market Watch            |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- número de símbolos mostrados en la ventana "Observación del mercado"
   int symbols=SymbolsTotal(true);
//--- posición del símbolo en la lista - un número aleatorio de 0 a symbols
   int number=MathRand()%symbols;
//--- devolvemos el nombre del símbolo sobre la posición indicada
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//|  cambia el color de segmentos de las velas
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- número de colores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- definimos de forma aleatoria un color nuevo para cada índice de colores
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- obtendremos un número aleatorio
      int number=MathRand();
      //--- obtendremos un índice en el array col[] como el remanente de la división c
      int i=number%size;
      //--- estableceremos el color para cada índice como la propiedad PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  número del estilo gráfico
                          PLOT_LINE_COLOR,      //  identificador de la propiedad
                          plot_color_ind,       //  índice del color donde escribiremo
                          cols[i]);             //  nuevo color
      //--- apuntaremos los colores
      comm=comm+StringFormat("CandleColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
```

© 2000-2025, MetaQuotes Ltd.

```
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| cambia la apariencia de las velas                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- cadena para formar la información sobre las propiedades de las velas
   string comm="";
//--- apuntamos el nombre del símbolo
   comm="\r\n"+symbol+comm;
//--- mostraremos la información en el gráfico a través del comentario
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

Relación entre las propiedades de indicador y funciones correspondientes

Cada uno de los indicadores personalizados tiene una gran cantidad de propiedades una parte de las cuales es obligatoria y siempre se encuentra al principio de la descripción. Estas propiedades son las siguientes:

- indicación de la ventana para mostrar el indicador - indicator_separate_window o indicator_chart_window;
- cantidad de los búfers de indicador - indicator_buffers;
- cantidad de construcciones (representaciones) gráficas que muestra el indicador - indicator_plots.

Pero hay otras propiedades que pueden ser definidas a través de las directivas del preprocesador y mediante las funciones para la creación del indicador personalizado. En la tabla de abajo se enumeran estas propiedades y sus funciones correspondientes.

Directivas para las propiedades de subventana del indicador	Funciones del tipo IndicatorSet...()	Descripción de la propiedad ajustada de subventana
indicator_height	IndicatorSetInteger (INDICATOR_INDICATOR_HEIGHT, nHeight)	Valor del alto fijo de la subventana
indicator_minimum	IndicatorSetDouble (INDICATOR_MINIMUM, dMaxValue)	Valor mínimo del eje vertical
indicator_maximum	IndicatorSetDouble (INDICATOR_MAXIMUM, dMinValue)	Valor máximo del eje vertical
indicator_levelN	IndicatorSetDouble (INDICATOR_LEVELVALUE, N-1, nLevelValue)	Valor del eje vertical para el nivel N
no hay directiva del preprocesador	IndicatorSetString (INDICATOR_LEVELTEXT, N-1, sLevelName)	Nombre del nivel mostrado
indicator_levelcolor	IndicatorSetInteger (INDICATOR_LEVELCOLOR, N-1, nLevelColor)	Color del nivel N
indicator_levelwidth	IndicatorSetInteger (INDICATOR_LEVELWIDTH, N-1, nLevelWidth)	Grosor de línea del nivel N
indicator_levelstyle	IndicatorSetInteger (INDICATOR_LEVELSTYLE, N-1, nLevelStyle)	Estilo de línea del nivel N
Directrices para las propiedades de	**Funciones del tipo PlotIndexSet...()**	**Descripción de la propiedad establecida para una**

© 2000-2025, MetaQuotes Ltd.

Directivas para las propiedades de subventana del indicador	Funciones del tipo IndicatorSet...()	Descripción de la propiedad ajustada de subventana
representaciones gráficas		representación gráfica
indicator_labelN	PlotIndexSetString(N-1,PLOT_LABEL,sLabel)	Denominación breve para representación gráfica N. Se muestra en la ventana DataWindow y en la ayuda contextual al apuntar con el cursor en el gráfico
indicator_colorN	PlotIndexSetInteger(N-1, PLOT_LINE_COLOR, nColor)	Color de línea para representación gráfica N
indicator_styleN	PlotIndexSetInteger(N-1, PLOT_LINE_STYLE, nType)	Estilo de línea para representación gráfica N
indicator_typeN	PlotIndexSetInteger(N-1, PLOT_DRAW_TYPE, nType)	Tipo de línea para representación gráfica N
indicator_widthN	PlotIndexSetInteger(N-1, PLOT_LINE_WIDTH, nWidth)	Grosor de línea para representación gráfica N
Propiedades generales de indicador	**Funciones del tipo IndicatorSet...()**	**Descripción**
no hay directiva del preprocesador	IndicatorSetString (INDICATOR_SHORTNAME, sShortName)	Establece un nombre breve y cómodo del indicador que va a visualizarse en la lista de indicadores (se llama en el terminal con la combinación **Ctrl+I**).
no hay directiva del preprocesador	IndicatorSetInteger (INDICATOR_DIGITS, nDigits)	Establece la precisión necesaria para visualizar valores del indicador, es decir, el número de dígitos después del punto decimal
no hay directiva del preprocesador	IndicatorSetInteger (INDICATOR_LEVELS, nLevels)	Establece la cantidad de niveles en la ventana del indicador
indicator_applied_price	No hay función, la propiedad se establece sólo con la directriz del preprocesador.	Tipo de precio por defecto que se utiliza para calcular el valor del indicador. Se especifica si hace falta sólo si se utiliza la función del primer tipo OnCalculate(). Valor de la propiedad puede ser establecido desde el diálogo de propiedades del

© 2000-2025, MetaQuotes Ltd.

Directivas para las propiedades de subventana del indicador	Funciones del tipo IndicatorSet...()	Descripción de la propiedad ajustada de subventana
		indicador en la pestaña "Parámetros" - "Aplicar a".

Cabe destacar que la numeración de niveles y representaciones gráficas en términos de preprocesador se empieza desde uno, mientras que la numeración de las mismas propiedades cuando se usan las funciones se empieza desde cero, es decir, hay que indicar el valor 1 menos que indicaríamos utilizando #property.

Hay algunas directrices para las que no existen funciones correspondientes:

Directriz	Descripción
indicator_chart_window	Indicador se visualiza en la ventana principal
indicator_separate_window	Indicador se visualiza en la subventana separada
indicator_buffers	Indica la cantidad necesaria de buffers requeridos
indicator_plots	Indica la cantidad de representaciones gráficas en el indicador

© 2000-2025, MetaQuotes Ltd.

SetIndexBuffer

Enlaza el búfer de indicador especificado con el array dinámico unidimensional del tipo double.

```
bool  SetIndexBuffer(
   int                   index,          // índice del buffer
   double                buffer[],       // array
   ENUM_INDEXBUFFER_TYPE  data_type      // lo que vamos a almacenar
   );
```

Parámetros

index

[in] Número del búfer de indicadores. La numeración se empieza desde 0. El número tiene que ser menos que el valor declarado en #property indicator_buffers.

buffer[]

[in] Array declarado en el programa del indicador personalizado.

data_type

[in] Tipo de datos guardados en el array de indicador. Por defecto INDICATOR_DATA (valores del indicador calculado). También puede adquirir el valor INDICATOR_COLOR_INDEX, entonces este búfer sirve para almacenar los índices de colores para el anterior búfer de indicadores. Se puede establecer hasta 64 colores en la línea #property indicator_colorN. El valor INDICATOR_CALCULATIONS significa que este buffer se usa en los cálculos intermedios del indicador y no está destinado para dibujar.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false.

Nota

Después de enlazado el array dinámico *buffer[]* tendrá la indexación como los arrays comunes, incluso si para este array previamente ha sido establecida la indexación como en las series temporales. Si hace falta cambiar el orden de acceso a los elementos del array de indicador, es necesario usar la función ArraySetAsSeries() después de enlazar el array usando la función SetIndexBuffer(). Además, hay que tener en cuenta que no se puede cambiar el tamaño de los arrays dinámicos que han sido asignados como búfers de indicadores por la función SetIndexBuffer(). Todas las operaciones relacionadas con el cambio de tamaño de los búfers de indicadores se realizan por el subsistema ejecutivo del terminal.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                TestCopyBuffer1.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input bool               AsSeries=true;
input int                period=15;
input ENUM_MA_METHOD     smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
input int                shift=0;
//--- indicator buffers
double                   MABuffer[];
int                      ma_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   if(AsSeries) ArraySetAsSeries(MABuffer,true);
   Print("Búfer de indicadores es una serie temporal = ",ArrayGetAsSeries(MABuffer));
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Búfer de indicadores después de SetIndexBuffer() es una serie temporal = ",
         ArrayGetAsSeries(MABuffer));

//--- vamos a cambiar el orden de acceso a los elementos del búfer de indicadores
   ArraySetAsSeries(MABuffer,AsSeries);

   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const long &volume[],
                const int &spread[])
  {
//--- vamos a copiar los valores del promedio móvil en el buffer MABuffer
   int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

   Print("MABuffer[0] = ",MABuffer[0]);// dependiendo del valor AsSeries
                                       // vamos a recibir el valor más antiguo
                                       // o en la barra actual no finalizada

//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Véase también

Propiedades de indicadores personalizados, Acceso a las series temporales e indicadores

© 2000-2025, MetaQuotes Ltd.

IndicatorSetDouble

Establece el valor de la propiedad correspondiente del indicador. La propiedad del indicador tiene que ser del tipo double. Existen 2 variantes de la función.

Llamada con especificación del identificador de la propiedad.

```
bool  IndicatorSetDouble(
   int     prop_id,           // identificador
   double  prop_value         // valor que se establece
   );
```

Llamada con especificación del identificador y modificador de la propiedad.

```
bool  IndicatorSetDouble(
   int     prop_id,           // identificador
   int     prop_modifier,     // modificador
   double  prop_value         // valor que se establece
   )
```

Parámetros

prop_id

[in] Identificador de la propiedad del indicador. Su valor puede ser uno de los valores de la enumeración ENUM_CUSTOMIND_PROPERTY_DOUBLE.

prop_modifier

[in] Modificador de la propiedad especificada. Sólo las propiedades de niveles requieren un modificador. La numeración de los niveles se empieza desde 0. Eso quiere decir que hay que especificar un uno para establecer la propiedad para el segundo nivel (uno menos que cuando se utiliza una directiva del compilador).

prop_value

[in] Valor de la propiedad.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false.

Nota

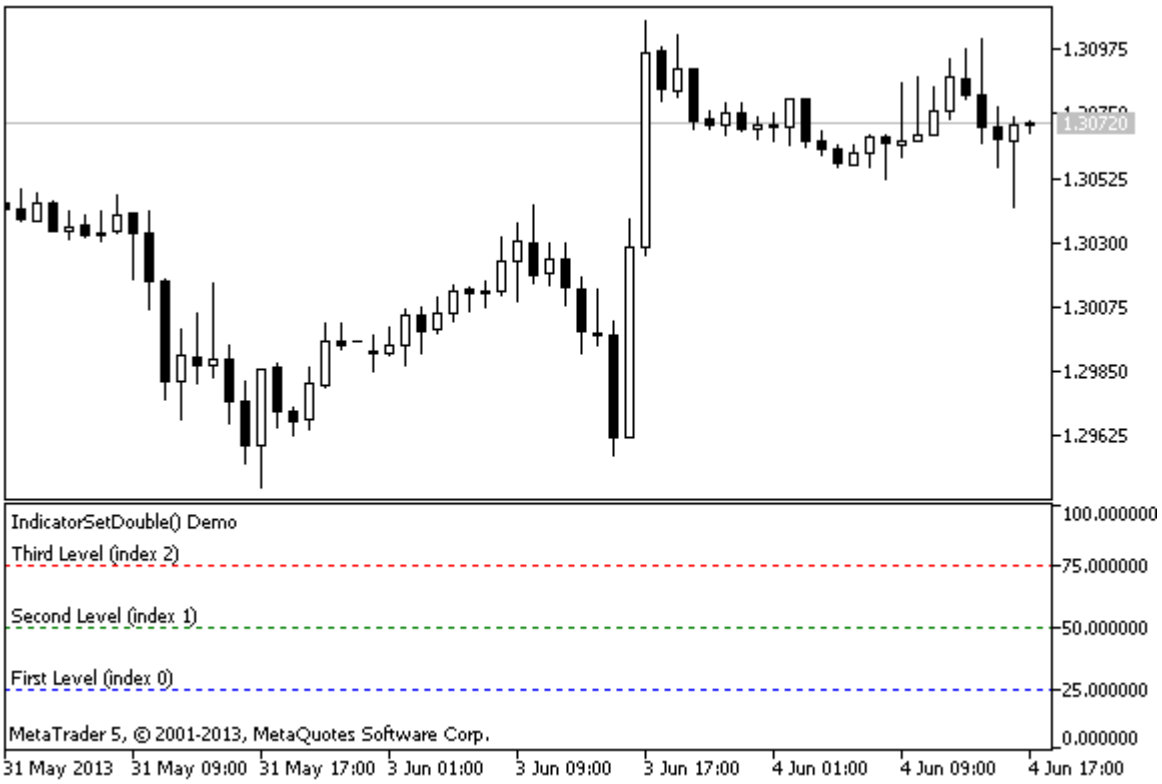
Cuando se utiliza la directiva #property la numeración de las propiedades (modificadores) se empieza desde 1 (un uno). Mientras que la función utiliza la numeración desde 0 (cero). Si el número del nivel no se establece de forma correcta, la visualización del indicador puede ser diferente de lo que está previsto.

Por ejemplo, hay dos maneras de establecer el valor del primer nivel para el indicador en una subventana separada:

- property indicator_level**1** 50 - se utiliza un 1 par especificar el número del nivel,
- IndicatorSetDouble(INDICATOR_LEVELVALUE, **0**, 50) - se utiliza 0 para especificar el primer nivel.

Ejemplo: indicador que cambia los valores máximos y mínimos de la ventana del indicador, así como los valores de los niveles en los que se ubican las líneas horizontales.

© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

```
//--- establecemos el valor máximo y mínimo para la ventana del indicador
#property indicator_minimum  0
#property indicator_maximum  100
//--- establecemos la muestra de tres niveles horizontales en la ventana separada del
#property indicator_level1 25
#property indicator_level2 50
#property indicator_level3 75
//--- establecemos el grosor de niveles horizontales
#property indicator_levelwidth 1
//--- establecemos el estilo de niveles horizontales
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ponemos las descripciones de niveles horizontales
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- ponemos el nombre breve al indicador
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetDouble() Demo");
//--- establecemos su color para cada nivel
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,0,clrBlue);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,1,clrGreen);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,2,clrRed);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
   static double level1=25,level2=50,level3=75;
   static double max=100,min=0, shift=100;
//--- contamos los ticks
   tick_counter++;
//--- en cada 10 tick hacemos la vuelta
   if(tick_counter%10==0)
     {
      //--- damos la vuelta al signo para los valores del nivel
      level1=-level1;
      level2=-level2;
      level3=-level3;
      //--- damos la vuelta al signo para los valores del máximo y mínimo
      max-=shift;
      min-=shift;
      //--- damos la vuelta al valor del desplazamiento
      shift=-shift;
      //--- establecemos nuevos valores de los niveles
```

© 2000-2025, MetaQuotes Ltd.

```
      IndicatorSetDouble(INDICATOR_LEVELVALUE,0,level1);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,1,level2);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,2,level3);
      //--- establecemos nuevos valores del máximo y mínimo para la ventana del indica
      Print("Set up max = ",max,",   min = ",min);
      IndicatorSetDouble(INDICATOR_MAXIMUM,max);
      IndicatorSetDouble(INDICATOR_MINIMUM,min);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Estilos de indicadores en ejemplos, Relación entre propiedades de indicador y funciones, Estilos de dibujo

© 2000-2025, MetaQuotes Ltd.

IndicatorSetInteger

Establece el valor de la propiedad correspondiente del indicador. La propiedad del indicador tiene que ser del tipo int o color. Existen 2 variantes de la función.

Llamada con especificación del identificador de la propiedad.

```
bool  IndicatorSetInteger(
   int  prop_id,           // identificador
   int  prop_value         // valor que se establece
   );
```

Llamada con especificación del identificador y modificador de la propiedad.

```
bool  IndicatorSetInteger(
   int  prop_id,           // identificador
   int  prop_modifier,     // modificador
   int  prop_value         // valor que se establece
   )
```

Parámetros

prop_id

[in] Identificador de la propiedad del indicador. Su valor puede ser uno de los valores de la enumeración ENUM_CUSTOMIND_PROPERTY_INTEGER.

prop_modifier

[in] Modificador de la propiedad especificada. Sólo las propiedades de niveles requieren un modificador.

prop_value

[in] Valor de la propiedad.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false.

Nota

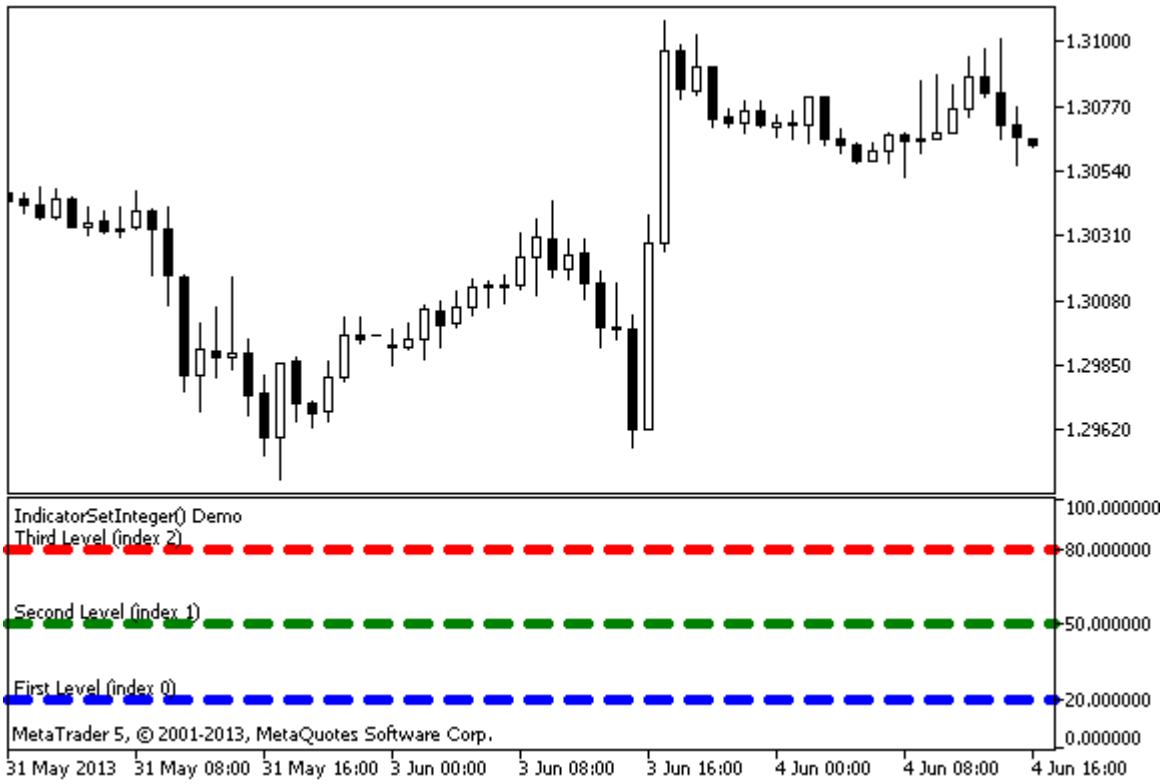
Cuando se utiliza la directiva #property la numeración de las propiedades (modificadores) se empieza desde 1 (un uno). Mientras que la función utiliza la numeración desde 0 (cero). Si el número del nivel no se establece de forma correcta, la visualización del indicador puede ser diferente de lo que está previsto.

Por ejemplo, para establecer el grosor de la línea del primer nivel horizontal utilice el índice cero:

- IndicatorSetInteger(INDICATOR_LEVELWIDTH, **0**, 5) - utilice índice 0 para establecer el grosor de la línea del primer nivel.

Ejemplo: indicador que establece el color, estilo y el grosor de los niveles horizontales del indicador.

© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- establecemos la muestra de tres niveles horizontales en la ventana separada del
#property indicator_level1 20
#property indicator_level2 50
#property indicator_level3 80
//--- establecemos el grosor de niveles horizontales
#property indicator_levelwidth 5
//--- establecemos el color de niveles horizontales
#property indicator_levelcolor clrAliceBlue
//--- establecemos el estilo de niveles horizontales
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ponemos las descripciones de niveles horizontales
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- ponemos el nombre breve al indicador
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetInteger() Demo");
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
//--- contamos los ticks
   tick_counter++;
//--- y establecemos los colores de niveles horizontales en función del contador de lo
   ChangeLevelColor(0,tick_counter,3,6,10); // tres últimos parámetros cambian el colo
   ChangeLevelColor(1,tick_counter,3,6,8);
   ChangeLevelColor(2,tick_counter,4,7,9);
//--- cambiamos los estilos de niveles horizontales
   ChangeLevelStyle(0,tick_counter);
   ChangeLevelStyle(1,tick_counter+5);
   ChangeLevelStyle(2,tick_counter+15);
//--- obtenemos el grosor de la línea como el resto de la división entera del número c
   int width=tick_counter%5;
//--- pasaremos por todos los niveles horizontales y pondremos
   for(int l=0;l<3;l++)
      IndicatorSetInteger(INDICATOR_LEVELWIDTH,l,width+1);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Establecemos el color de la línea horizontal en la ventana separada del indicador
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeLevelColor(int level,      // número de la línea horizontal
                      int tick_number,// el dividiendo, número para obtener el resto d
                      int f_trigger,  // el primer divisor de cambio del color
                      int s_trigger,  // el segundo divisor de cambio del color
                      int t_trigger)  // el tercer divisor de cambio del color
  {
   static color colors[3]={clrRed,clrBlue,clrGreen};
//--- índice del color desde el array colors[]
   int index=-1;
//--- calculamos el número del color desde el colors[] para colorear la línea horizont
   if(tick_number%f_trigger==0)
      index=0;   // si el número tick_number se divide sin resto por f_trigger
   if(tick_number%s_trigger==0)
      index=1;   // si el número tick_number se divide sin resto por s_trigger
   if(tick_number%t_trigger==0)
      index=2;   // si el número tick_number se divide sin resto por t_trigger
//--- si el color está determinado, lo establecemos
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELCOLOR,level,colors[index]);
//---
  }
//+------------------------------------------------------------------+
//| Establecemos el estilo de la línea horizontal en la ventana separada del indicador
//+------------------------------------------------------------------+
void ChangeLevelStyle(int level,     // número de la línea horizontal
                      int tick_number// número para obtener el resto de la división)
                      )
  {
//--- array para guardar los estilos
   static ENUM_LINE_STYLE styles[5]=
     {STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//--- índice del estilo desde el array styles[]
   int index=-1;
//--- calculamos el número desde el array styles[] para establecer el estilo de la lín
   if(tick_number%50==0)
      index=5;   // si tick_number se divide sin resto por 50, el estilo será STYLE_DA
   if(tick_number%40==0)
      index=4;   // ... estilo STYLE_DASHDOT
   if(tick_number%30==0)
      index=3;   // ... STYLE_DOT
   if(tick_number%20==0)
      index=2;   // ... STYLE_DASH
   if(tick_number%10==0)
      index=1;   // ... STYLE_SOLID
//--- si el estilo está determinado, lo establecemos
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELSTYLE,level,styles[index]);
  }
```

Véase también

Propiedades de indicadores personalizados, Propiedades de programas (#property), Estilos de dibujo

© 2000-2025, MetaQuotes Ltd.

IndicatorSetString

Establece el valor de la propiedad correspondiente del indicador. La propiedad del indicador tiene que ser del tipo string. Existen 2 variantes de la función.

Llamada con especificación del identificador de la propiedad.

```
bool  IndicatorSetString(
   int     prop_id,           // identificador
   string  prop_value         // valor que se establece
   );
```

Llamada con especificación del identificador y modificador de la propiedad.

```
bool  IndicatorSetString(
   int     prop_id,           // identificador
   int     prop_modifier,     // modificador
   string  prop_value         // valor que se establece
   )
```

Parámetros

prop_id

[in] Identificador de la propiedad del indicador. Su valor puede ser uno de los valores de la enumeración ENUM_CUSTOMIND_PROPERTY_STRING.

prop_modifier

[in] Modificador de la propiedad especificada. Sólo las propiedades de niveles requieren un modificador.

prop_value

[in] Valor de la propiedad.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false.

Nota

Cuando se utiliza la directiva #property la numeración de las propiedades (modificadores) se empieza desde 1 (un uno). Mientras que la función utiliza la numeración desde 0 (cero). Si el número del nivel no se establece de forma correcta, la visualización del indicador puede ser diferente de lo que está previsto.

Por ejemplo, para establecer la descripción del primer nivel horizontal utilice el índice cero:

- IndicatorSetString(INDICATOR_LEVELTEXT, **0**, "First Level") - utilice el índice 0 para establecer la descripción de texto del primer nivel.

Ejemplo: indicador que establece el texto para los niveles horizontales del indicador.

© 2000-2025, MetaQuotes Ltd.




```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- establecemos la muestra de tres niveles horizontales en la ventana separada del
#property indicator_level1 30
#property indicator_level2 50
#property indicator_level3 70
//--- establecemos el color de niveles horizontales
#property indicator_levelcolor clrRed
//--- establecemos el estilo de niveles horizontales
#property indicator_levelstyle STYLE_SOLID
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- establecemos las descripciones de niveles horizontales
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- ponemos el nombre breve al indicador
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetString() Demo");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---

//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Véase también

Propiedades de indicadores personalizados, Propiedades de programas (#property)

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetDouble

Establece el valor de la propiedad correspondiente de línea correspondiente del indicador. La propiedad del indicador tiene que ser del tipo double.

```
bool  PlotIndexSetDouble(
   int     plot_index,     // índice del estilo gráfico
   int     prop_id,        // identificador de la propiedad
   double  prop_value      // valor que se establece
   );
```

Parámetros

plot_index

[in] Índice de la representación gráfica

prop_id

[in] Identificador de la propiedad del indicador. Su valor puede ser uno de los valores de la enumeración ENUM_PLOT_PROPERTY_DOUBLE.

prop_value

[in] Valor de la propiedad.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetInteger

Establece el valor de la propiedad correspondiente de línea correspondiente del indicador. La propiedad del indicador tiene que ser del tipo int, char, bool o color. Existen 2 variantes de la función.

Llamada con especificación del identificador de la propiedad.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // índice del estilo gráfico
   int  prop_id,            // identificador de la propiedad
   int  prop_value          // valor que se establece
   );
```

Llamada con especificación del identificador y modificador de la propiedad.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // índice del estilo gráfico
   int  prop_id,            // identificador de la propiedad
   int  prop_modifier,      // modificador de la propiedad
   int  prop_value          // valor que se establece
   )
```

Parámetros

plot_index

[in] Índice de la representación gráfica

prop_id

[in] Identificador de la propiedad del indicador. Su valor puede ser uno de los valores de la enumeración ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Modificador de la propiedad especificada. Sólo las propiedades de los índices de colores requieren un modificador.

prop_value

[in] Valor de la propiedad.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false.

Ejemplo: indicador que dibuja la línea de tres colores. El esquema de colores se cambia cada 5 ticks.

© 2000-2025, MetaQuotes Ltd.




```
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plot ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrGreen,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- indicator buffers
double         ColorLineBuffer[];
double         ColorBuffer[];
int            MA_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- get MA handle
   MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| get color index                                                  |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int getIndexOfColor(int i)
  {
   int j=i%300;
   if(j<100) return(0);// first index
   if(j<200) return(1);// second index
   return(2); // third index
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   static int ticks=0,modified=0;
   int limit;
//--- first calculation or number of bars was changed
   if(prev_calculated==0)
     {
      //--- copy values of MA into indicator buffer ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);// copying failed - throw away
      //--- now set line color for every bar
      for(int i=0;i<rates_total;i++)
         ColorBuffer[i]=getIndexOfColor(i);
     }
   else
     {
      //--- copy values of MA into indicator buffer ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);

      ticks++;// ticks counting
      if(ticks>=5)//it's time to change color scheme
        {
         ticks=0; // reset counter
         modified++; // counter of color changes
         if(modified>=3)modified=0;// reset counter
         ResetLastError();
         switch(modified)
           {
```

© 2000-2025, MetaQuotes Ltd.

```
            case 0:// first color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
               Print("Color scheme "+modified);
               break;
            case 1:// second color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
               Print("Color scheme "+modified);
               break;
            default:// third color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
               Print("Color scheme "+modified);
           }
        }
      else
        {
         //--- set start position
         limit=prev_calculated-1;
         //--- now we set line color for every bar
         for(int i=limit;i<rates_total;i++)
            ColorBuffer[i]=getIndexOfColor(i);
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetString

Establece el valor de la propiedad correspondiente de línea correspondiente del indicador. La propiedad del indicador tiene que ser del tipo string.

```
bool  PlotIndexSetString(
   int     plot_index,     // índice del estilo gráfico
   int     prop_id,        // identificador de la propiedad
   string  prop_value      // valor que se establece
   );
```

Parámetros

plot_index

[in] Índice de la representación gráfica

prop_id

[in] Identificador de la propiedad del indicador. Su valor puede ser uno de los valores de la enumeración ENUM_PLOT_PROPERTY_STRING.

prop_value

[in] Valor de la propiedad.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

PlotIndexGetInteger

Devuelve el valor de la propiedad correspondiente de línea correspondiente del indicador. La propiedad del indicador tiene que ser del tipo int, color, bool o char. Existen 2 variantes de la función.

Llamada con especificación del identificador de la propiedad.

```
int  PlotIndexGetInteger(
   int  plot_index,         // índice del estilo gráfico
   int  prop_id,            // identificador de la propiedad
   );
```

Llamada con especificación del identificador y modificador de la propiedad.

```
int  PlotIndexGetInteger(
   int  plot_index,         // índice del estilo gráfico
   int  prop_id,            // identificador de la propiedad
   int  prop_modifier       // modificador de la propiedad
   )
```

Parámetros

plot_index

[in] Índice de construcción gráfica

prop_id

[in] Identificador de la propiedad del indicador. Su valor puede ser uno de los valores de la enumeración ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Modificador de la propiedad especificada. Sólo las propiedades de los índices de colores requieren un modificador.

Nota

La función está destinada para extraer las configuraciones de dibujo de una línea correspondiente del indicador. Esta función trabaja junto con la función PlotIndexSetInteger teniendo objetivo de copiar las propiedades de dibujo de una línea en otra.

Ejemplo: El indicador que pinta las velas de un color que depende del día de la semana. Los colores para cada día se establecen de un modo programado.

© 2000-2025, MetaQuotes Ltd.




```
#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double          OpenBuffer[];
double          HighBuffer[];
double          LowBuffer[];
double          CloseBuffer[];
double          ColorCandlesColors[];
color           ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                               clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,OpenBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,HighBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LowBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,CloseBuffer,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- set number of colors in color buffer
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- set colors for color buffer
   for(int i=1;i<6;i++)
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- set accuracy
   IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
   printf("We have %u colors of days",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   int i;
   MqlDateTime t;
//----
   if(prev_calculated==0) i=0;
   else i=prev_calculated-1;
//----
   while(i<rates_total)
     {
      OpenBuffer[i]=open[i];
      HighBuffer[i]=high[i];
      LowBuffer[i]=low[i];
      CloseBuffer[i]=close[i];
      //--- set color for every candle
      TimeToStruct(time[i],t);
      ColorCandlesColors[i]=t.day_of_week;
      //---
      i++;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

Objetos gráficos

Es el grupo de funciones destinadas para trabajar con objetos gráficos pertenecientes a cualquier gráfico especificado. No se puede usar estas funciones en los indicadores.

Las funciones que establecen las propiedades de objetos gráficos, así como las operaciones de creación ObjectCreate() y movimiento ObjectMove() de los objetos en el gráfico, sirven prácticamente para mandar los comandos al gráfico. Cuando estas funciones se ejecuten con éxito, el comando se coloca en la cola general de los eventos del gráfico. El cambio visual de los objetos gráficos se realiza durante el procesamiento de la cola de eventos de este gráfico.

Por esta razón no hay que esperar la modificación visual inmediata de los objetos gráficos tras la llamada de estas funciones. En general la actualización de los objetos gráficos se realiza por el terminal de forma automática según los eventos del cambio, es decir: la llegada de una nueva cotización, cambio del tamaño de la ventana, etc.

Para la actualización forzosa de los objetos gráficos, se utiliza el comando de redibujo del gráfico ChartRedraw().

Función	Acción
ObjectCreate	Crea un objeto del tipo especificado en un gráfico especificado
ObjectName	Devuelve el nombre de un objeto del tipo correspondiente en el gráfico especificado (subventana del gráfico especificada)
ObjectDelete	Elimina el objeto con el nombre especificado del gráfico especificado (subventana del gráfico especificada)
ObjectsDeleteAll	Elimina todos los objetos del tipo especificado del gráfico especificado (subventana del gráfico especificada)
ObjectFind	Busca por el nombre un objeto con el identificador especificado
ObjectGetTimeByValue	Devuelve el valor de hora para el valor especificado del precio de un objeto
ObjectGetValueByTime	Devuelve el valor de precio de un objeto para la hora especificada
ObjectMove	Cambia las coordinadas del punto de anclaje de un objeto
ObjectsTotal	Devuelve el número de objetos del tipo especificado en el gráfico especificado (subventana del gráfico especificada)
ObjectGetDouble	Devuelve el valor del tipo double de la propiedad correspondiente de un objeto
ObjectGetInteger	Devuelve el valor de números enteros de la propiedad correspondiente de un objeto
ObjectGetString	Devuelve el valor del tipo string de la propiedad correspondiente de un objeto
ObjectSetDouble	Establece el valor de propiedad correspondiente de un objeto
ObjectSetInteger	Establece el valor de propiedad correspondiente de un objeto

© 2000-2025, MetaQuotes Ltd.

Función	Acción
ObjectSetString	Establece el valor de propiedad correspondiente de un objeto
TextSetFont	Establece la fuente para visualizar el texto a través de los métodos del dibujo (por defecto se usa Arial 20)
TextOut	Pasa el texto al array personalizado (búfer) que sirve para crear un recurso gráfico
TextGetSize	Devuelve el alto y el ancho de la cadena con los ajustes de la fuente actuales

Cada objeto gráfico debe tener el nombre único dentro de los márgenes de un gráfico, incluyendo sus subventanas. El cambio del nombre de un objeto gráfico genera dos eventos: evento de eliminación del objeto con el nombre anterior, evento de creación del objeto gráfico con el nombre nuevo.

Después de crear un objeto o modificar las propiedades del objeto se recomienda llamar a la función ChartRedraw(), la que ordena al terminal dibujar el gráfico por vía forzada (y todos sus objetos visibles).

© 2000-2025, MetaQuotes Ltd.

ObjectCreate

Crea un objeto con el nombre especificado, tipo y coordinadas iniciales en la subventana del gráfico especificada. Durante la creación se puede indicar hasta 30 coordinadas.

```
bool  ObjectCreate(
   long          chart_id,       // identificador del gráfico
   string        name,           // nombre del objeto
   ENUM_OBJECT   type,           // tipo de objeto
   int           sub_window,     // índice de ventana
   datetime      time1,          // hora del primer punto de anclaje
   double        price1,         // precio del primer punto de anclaje
   ...
   datetime      timeN=0,        // hora de punto de anclaje N
   double        priceN=0,       // precio de punto de anclaje N
   ...
   datetime      time30=0,       // hora del punto de anclaje 30
   double        price30=0       // precio del punto de anclaje 30
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto. El nombre tiene que ser único dentro de los límites de un gráfico, incluyendo sus subventanas.

type

[in] Tipo de objeto. Su valor puede ser uno de los valores de la enumeración ENUM_OBJECT.

sub_window

[in] Número de subventana del gráfico. 0 significa la ventana principal del gráfico. La subventana especificada debe existir, de lo contrario la función devuelve false.

time1

[in] Coordinada de hora del primer enlace.

price1

[in] Coordinada de precio del primer punto de anclaje.

timeN=0

[in] Coordinada de hora del punto de anclaje N.

priceN=0

[in] Coordinada de precio del punto de anclaje N.

time30=0

[in] Coordinada de hora del punto de anclaje 30.

price30=0

© 2000-2025, MetaQuotes Ltd.

[in] Coordinada de precio del punto de anclaje 30.

Valor devuelto

Retorna true en el caso de que se haya añadido con éxito el comando a la cola del gráfico indicado, de lo contrario, false. Si el objeto ya ha sido creado con anterioridad, entonces se realizará un intento de cambiar sus coordenadas.

Nota

Al llamar ObjectCreate() siempre se usa una llamada asincrónica, por eso la función retorna solo el resultado de la colocación del comando en la cola del gráfico. En este caso, true solo significa que el comando se ha puesto en la cola con éxito, el propio resultado de su ejecución aún se desconoce.

Para comprobar el resultado de la ejecución se puede usar la función ObjectFind() o cualquier función que retorne las propiedades del objeto, por ejemplo, del tipo ObjectGetXXX. Pero, en esta caso, además, se deberá tener en cuenta que estas funciones se colocan al final de la cola de comandos del gráfico y esperan el resultado de la ejecución (puesto que que son llamadas sincrónicas), es decir, pueden consumir bastante tiempo. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

El nombre del objeto gráfico no debe superar 63 caracteres.

La numeración de las subventanas del gráfico (si en el gráfico hay subventanas con indicadores) se empieza desde 1. Siempre existe la venta principal y tiene el índice 0.

La gran cantidad de puntos de anclaje (hasta 30) está prevista para usarlas en el futuro. Al mismo tiempo el límite de 30 posibles puntos de anclaje para los objetos gráficos se debe al hecho que durante la llamada a la función el número de parámetros no debe superar 64.

Al renombrar un objeto gráfico, se generan al mismo tiempo dos eventos que pueden ser procesados en el Asesor Experto o el indicador usando la función OnChartEvent():

- evento de eliminación del objeto con el nombre anterior;
- evento de creación del objeto gráfico con el nombre nuevo.

Para crear cada uno de los siguientes tipos de objetos, es necesario fijar una cierta cantidad de los puntos de anclaje:

Identificador	Descripción	Puntos de anclaje
OBJ_VLINE	Línea vertical	Un punto de anclaje. Prácticamente se utiliza sólo la coordinada del eje de tiempo.
OBJ_HLINE	Línea horizontal	Un punto de anclaje. Prácticamente se utiliza sólo la coordinada del eje de precio.
OBJ_TREND	Línea de tendencia	Dos puntos de anclaje.
OBJ_TRENDBYANGLE	Línea de tendencia por ángulo	Dos puntos de anclaje.
OBJ_CYCLES	Líneas cíclicas	Dos puntos de anclaje.

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Puntos de anclaje
OBJ_ARROWED_LINE	Objeto "Línea con flecha"	Dos puntos de anclaje.
OBJ_CHANNEL	Canal equidistante	Tres puntos de anclaje.
OBJ_STDDEVCHANNEL	Canal de desviación estándar	Dos puntos de anclaje.
OBJ_REGRESSION	Canal de regresión lineal	Dos puntos de anclaje.
OBJ_PITCHFORK	Tridente de Andrews	Tres puntos de anclaje.
OBJ_GANNLINE	Línea de Gann	Dos puntos de anclaje.
OBJ_GANNFAN	Abanico de Gann	Dos puntos de anclaje.
OBJ_GANNGRID	Cuadrícula de Gann	Dos puntos de anclaje.
OBJ_FIBO	Niveles de Fibonacci	Dos puntos de anclaje.
OBJ_FIBOTIMES	Zonas temporales de Fibonacci	Dos puntos de anclaje.
OBJ_FIBOFAN	Abanico de Fibonacci	Dos puntos de anclaje.
OBJ_FIBOARC	Arcos de Fibonacci	Dos puntos de anclaje.
OBJ_FIBOCHANNEL	Canal de Fibonacci	Tres puntos de anclaje.
OBJ_EXPANSION	Expansión de Fibonacci	Tres puntos de anclaje.
OBJ_ELLIOTWAVE5	5 ondas de Elliott	Cinco puntos de anclaje.
OBJ_ELLIOTWAVE3	3 ondas de Elliott	Tres puntos de anclaje.
OBJ_RECTANGLE	Rectángulo	Dos puntos de anclaje.
OBJ_TRIANGLE	Triángulo	Tres puntos de anclaje.
OBJ_ELLIPSE	Elipse	Tres puntos de anclaje.
OBJ_ARROW_THUMB_UP	Signo "Bien" (pulgar arriba)	Un punto de anclaje.
OBJ_ARROW_THUMB_DOWN	Signo "Mal" (pulgar abajo)	Un punto de anclaje.
OBJ_ARROW_UP	Signo "Flecha arriba"	Un punto de anclaje.
OBJ_ARROW_DOWN	Signo "Flecha abajo"	Un punto de anclaje.
OBJ_ARROW_STOP	Signo "Stop"	Un punto de anclaje.
OBJ_ARROW_CHECK	Signo "Visto" (marca de comprobación)	Un punto de anclaje.
OBJ_ARROW_LEFT_PRICE	Etiqueta izquierda de precio	Un punto de anclaje.
OBJ_ARROW_RIGHT_PRICE	Etiqueta derecha de precio	Un punto de anclaje.
OBJ_ARROW_BUY	Signo "Buy"	Un punto de anclaje.
OBJ_ARROW_SELL	Signo "Sell"	Un punto de anclaje.

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Puntos de anclaje
OBJ_ARROW	Objeto "Flecha"	Un punto de anclaje.
OBJ_TEXT	Objeto "Texto"	Un punto de anclaje.
OBJ_LABEL	Objeto "Etiqueta de texto"	La posición se fija utilizando las propiedades OBJPROP_XDISTANCE y OBJPROP_YDISTANCE.
OBJ_BUTTON	Objeto "Botón"	La posición se fija utilizando las propiedades OBJPROP_XDISTANCE y OBJPROP_YDISTANCE.
OBJ_CHART	Objeto "Gráfico"	La posición se fija utilizando las propiedades OBJPROP_XDISTANCE y OBJPROP_YDISTANCE.
OBJ_BITMAP	Objeto "Imagen"	Un punto de anclaje.
OBJ_BITMAP_LABEL	Objeto "Etiqueta gráfica"	La posición se fija utilizando las propiedades OBJPROP_XDISTANCE y OBJPROP_YDISTANCE.
OBJ_EDIT	Objeto "Campo de edición"	La posición se fija utilizando las propiedades OBJPROP_XDISTANCE y OBJPROP_YDISTANCE.
OBJ_EVENT	Objeto "Evento" que corresponde a un evento en el calendario económico	Un punto de anclaje. Prácticamente se utiliza sólo la coordinada del eje de tiempo.
OBJ_RECTANGLE_LABEL	Objeto "Etiqueta rectangular" para crear y personalizar la interfaz gráfica de usuario.	La posición se fija utilizando las propiedades OBJPROP_XDISTANCE y OBJPROP_YDISTANCE.

© 2000-2025, MetaQuotes Ltd.

ObjectName

Devuelve el nombre del objeto correspondiente en el gráfico especificado (subventana del gráfico especificada) del tipo especificado.

```
string  ObjectName(
   long  chart_id,           // identificador del gráfico
   int   pos,                // número en la lista de objetos
   int   sub_window=-1,      // número de ventana
   int   type=-1             // tipo de objeto
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

pos

[in] Número ordinal del objeto de acuerdo con el filtro especificado según el número de subventana y tipo.

nwin=-1

[in] Número de subventana del gráfico. 0 significa la ventana principal, -1 significa todas las subventanas del gráfico, incluyendo la ventana principal.

type=-1

[in] Tipo del objeto. Su valor puede ser uno de los valores de la enumeración ENUM_OBJECT. -1 significa todos los tipos.

Valor devuelto

Nombre del objeto en caso del éxito.

Nota

La función usa una llamada sincrónica, esto significa que la función espera a la ejecución de todos los comandos que han sido ubicados en la cola del gráfico antes de su llamada, y por eso puede consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

Al renombrar un objeto gráfico, se generan al mismo tiempo dos eventos que pueden ser procesados en el Asesor Experto o el indicador usando la función OnChartEvent():

- evento de eliminación del objeto con el nombre anterior;
- evento de creación del objeto gráfico con el nombre nuevo.

© 2000-2025, MetaQuotes Ltd.

ObjectDelete

Elimina el objeto con el nombre especificado del gráfico especificado.

```
bool  ObjectDelete(
   long    chart_id,      // identificador del gráfico
   string  name           // nombre del objeto
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto a eliminar.

Valor devuelto

Retorna true en el caso de que se haya añadido con éxito el comando a la cola del gráfico indicado, de lo contrario, false.

Nota

Al llamar ObjectDelete() siempre se usa una llamada asincrónica, por eso la función retorna solo el resultado de la colocación de la orden en la cola del gráfico. En este caso, true solo significa que el comando se ha puesto en la cola con éxito, el propio resultado de su ejecución aún se desconoce.

Para comprobar el resultado de la ejecución se puede usar la función ObjectFind() o cualquier función que retorne las propiedades del objeto, por ejemplo, del tipo ObjectGetXXX. Pero, en esta caso, además, se deberá tener en cuenta que estas funciones se colocan al final de la cola de comandos del gráfico y esperan el resultado de la ejecución (puesto que que son llamadas sincrónicas), es decir, pueden consumir bastante tiempo. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

Al renombrar un objeto gráfico, se generan al mismo tiempo dos eventos que pueden ser procesados en el Asesor Experto o el indicador usando la función OnChartEvent():

- evento de eliminación del objeto con el nombre anterior;
- evento de creación del objeto gráfico con el nombre nuevo.

© 2000-2025, MetaQuotes Ltd.

ObjectsDeleteAll

Elimina todos los objetos del tipo especificado del gráfico especificado (subventana del gráfico especificada).

```
int  ObjectsDeleteAll(
   long  chart_id,           // identificador del gráfico
   int   sub_window=-1,      // índice de ventana
   int   type=-1             // tipo del objeto a eliminar
   );
```

Elimina todos los objetos del tipo especificado en la subventana del gráfico según el prefijo del nombre.

```
int  ObjectsDeleteAll(
   long           chart_id,  // identificador del gráfico
   const string     prefix,  // prefijo del nombre del objeto
   int        sub_window=-1,  // índice de la ventana
   int       object_type=-1   // tipo del objeto a eliminar
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

prefix

[in] Prefijo para eliminar todos los objetos cuyos nombres empiezan con este conjunto de caracteres. Se puede indicar el perfijo como 'name' o 'name*' - ambas opciones funcionan igualmente. Si se indica la línea vacía como prefijo, se eliminarán los objetos con cualquier nombre.

sub_window=-1

[in] Número de subventana del gráfico. 0 significa la ventana principal, -1 significa todas las subventanas del gráfico, incluyendo la ventana principal.

type=-1

[in] Tipo del objeto. Su valor puede ser uno de los valores de la enumeración ENUM_OBJECT. -1 significa todos los tipos.

Valor devuelto

Devuelve el número de objetos eliminados. Para la información más detallada sobre el error hay que llamar a la función GetLastError().

Nota

La función usa una llamada sincrónica, esto significa que la función espera a la ejecución de todos los comandos que han sido ubicados en la cola del gráfico antes de su llamada, y por eso puede consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

© 2000-2025, MetaQuotes Ltd.

ObjectFind

Busca un objeto con el nombre especificado en el gráfico con el identificador especificado.

```
int  ObjectFind(
   long    chart_id,      // identificador del gráfico
   string  name           // nombre del objeto
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto que se busca.

Valor devuelto

En caso del éxito la función devuelve el número de subventana (0 significa la ventana principal del gráfico) donde se encuentra el objeto encontrado. Si el objeto no ha sido encontrado, la función devuelve un número negativo. Para la información más detallada sobre el error hay que llamar a la función GetLastError().

Nota

La función usa una llamada sincrónica, esto significa que la función espera a la ejecución de todos los comandos que han sido ubicados en la cola del gráfico antes de su llamada, y por eso puede consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

Al renombrar un objeto gráfico, se generan al mismo tiempo dos eventos que pueden ser procesados en el Asesor Experto o el indicador usando la función OnChartEvent():

- evento de eliminación del objeto con el nombre anterior;
- evento de creación del objeto gráfico con el nombre nuevo.

© 2000-2025, MetaQuotes Ltd.

ObjectGetTimeByValue

Devuelve el valor de la hora para el valor especificado del precio de un objeto especificado.

```
datetime  ObjectGetTimeByValue(
   long     chart_id,      // identificador del gráfico
   string   name,          // nombre del objeto
   double   value,         // precio
   int      line_id        // línea
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto.

value

[in] Valor del precio.

line_id

[in] Identificador de la línea.

Valor devuelto

Devuelve el valor de la hora para el valor especificado del precio de un objeto especificado.

Nota

La función usa una llamada sincrónica, esto significa que la función espera a la ejecución de todos los comandos que han sido ubicados en la cola del gráfico antes de su llamada, y por eso puede consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

Puesto que el objeto puede tener varios valores en una coordinada de precio, en este caso es necesario especificar el indicador de la línea. Esta función se puede aplicar sólo a los siguientes objetos:

- Línea de tendencia (OBJ_TREND)
- Línea de tendencia por ángulo (OBJ_TRENDBYANGLE)
- Línea de Gann (OBJ_GANNLINE)
- Canal equidistante (OBJ_CHANNEL) - 2 líneas
- Canal de regresión lineal (OBJ_REGRESSION) - 3 líneas
- Canal de desviación estándar (OBJ_STDDEVCHANNEL) - 3 líneas
- Flecha (OBJ_ARROWED_LINE)

Véase también

Tipos de objetos

© 2000-2025, MetaQuotes Ltd.

ObjectGetValueByTime

Devuelve el valor de precio para la hora especificada de un objeto especificado.

```
double  ObjectGetValueByTime(
   long      chart_id,     // identificador del gráfico
   string    name,         // nombre del objeto
   datetime  time,         // hora
   int       line_id       // línea
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto.

time

[in] Valor de la hora.

line_id

[in] Identificador de la línea.

Valor devuelto

Devuelve el valor de precio para la hora especificada de un objeto especificado.

Nota

La función usa una llamada sincrónica, esto significa que la función espera a la ejecución de todos los comandos que han sido ubicados en la cola del gráfico antes de su llamada, y por eso puede consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

Puesto que el objeto puede tener varios valores en una coordinada de precio, en este caso es necesario especificar el indicador de la línea. Esta función se puede aplicar sólo a los siguientes objetos:

- Línea de tendencia (OBJ_TREND)
- Línea de tendencia por ángulo (OBJ_TRENDBYANGLE)
- Línea de Gann (OBJ_GANNLINE)
- Canal equidistante (OBJ_CHANNEL) - 2 líneas
- Canal de regresión lineal (OBJ_REGRESSION) - 3 líneas
- Canal de desviación estándar (OBJ_STDDEVCHANNEL) - 3 líneas
- Flecha (OBJ_ARROWED_LINE)

Véase también

Tipos de objetos

© 2000-2025, MetaQuotes Ltd.

ObjectMove

Cambia las coordinadas del punto de anclaje de un objeto.

```
bool  ObjectMove(
   long      chart_id,         // identificador del gráfico
   string    name,             // nombre del objeto
   int       point_index,      // número de anclaje
   datetime  time,             // hora
   double    price             // precio
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto.

point_index

[in] Número del punto de anclaje. La cantidad de puntos de anclaje depende del tipo de objeto.

time

[in] Coordinada de hora del punto de anclaje especificado.

price

[in] Coordinada de precio del punto de anclaje especificado.

Valor devuelto

Retorna true en el caso de que se haya añadido con éxito el comando a la cola del gráfico indicado, de lo contrario, false.

Nota

Al llamar ObjectMove() siempre se usa una llamada asincrónica, por eso la función retorna solo el resultado de la colocación de la orden en la cola del gráfico. En este caso, true solo significa que el comando se ha puesto en la cola con éxito, el propio resultado de su ejecución aún se desconoce.

Para comprobar el resultado de la ejecución se puede usar una función que solicite las propiedades del objeto, por ejemplo, del tipo ObjectGetXXX. Pero, en esta caso, además, se deberá tener en cuenta que estas funciones se colocan al final de la cola de comandos del gráfico y esperan el resultado de la ejecución (puesto que que son llamadas sincrónicas), es decir, pueden consumir bastante tiempo. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

© 2000-2025, MetaQuotes Ltd.

ObjectsTotal

Devuelve el número de objetos del tipo especificado en el gráfico especificado (subventana del gráfico especificada).

```
int  ObjectsTotal(
   long  chart_id,           // identificador del gráfico
   int   sub_window=-1,      // índice de ventana
   int   type=-1             // tipo de objeto
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

nwin=-1

[in] Número de subventana del gráfico. 0 significa la ventana principal, -1 significa todas las subventanas del gráfico, incluyendo la ventana principal.

type=-1

[in] Tipo del objeto. Su valor puede ser uno de los valores de la enumeración ENUM_OBJECT. -1 significa todos los tipos.

Valor devuelto

Número de objetos.

Nota

La función usa una llamada sincrónica, esto significa que la función espera a la ejecución de todos los comandos que han sido ubicados en la cola del gráfico antes de su llamada, y por eso puede consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

© 2000-2025, MetaQuotes Ltd.

ObjectSetDouble

Establece el valor de la propiedad correspondiente de un objeto. La propiedad del objeto debe ser del tipo double. Existen 2 variantes de la función.

Configuración del valor de la propiedad sin modificador

```
bool  ObjectSetDouble(
   long                          chart_id,          // identificador del gráfico
   string                        name,              // nombre
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // propiedad
   double                        prop_value         // valor
   );
```

Configuración del valor de la propiedad indicando el modificador

```
bool  ObjectSetDouble(
   long                          chart_id,          // identificador del gráfico
   string                        name,              // nombre
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // propiedad
   int                           prop_modifier,     // modificador
   double                        prop_value         // valor
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto.

prop_id

[in] Identificador de la propiedad del objeto. Su valor puede ser uno de los valores de la enumeración ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Modificador de la propiedad especificada. La mayoría de las propiedades no requiere el modificador.

prop_value

[in] Valor de la propiedad.

Valor devuelto

Devuelve true sólo si el comando de modificar las propiedades de un objeto gráfico se ha enviado con éxito al gráfico. De lo contrario devuelve false. Para la información más detallada sobre el error hay que llamar a la función GetLastError().

Nota

© 2000-2025, MetaQuotes Ltd.

La función usa una llamada asincrónica, esto significa que la función no espera a la ejecución del comando que se ha colocado con éxito en la cola del gráfico indicado, sino que devuelve el control de inmediato.

Para comprobar el resultado de la ejecución en un gráfico ajeno se puede usar una función que solicite la propiedad indicada del objeto. Pero, en esta caso, además, se deberá tener en cuenta que estas funciones se colocan al final de la cola de comandos del gráfico y esperan el resultado de la ejecución, es decir, pueden consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

Ejemplo de creación de un objeto Fibonacci y adicción de un nuevo nivel en él

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- matrices auxiliares
   double high[],low[],price1,price2;
   datetime time[],time1,time2;
//--- vamos a copiar los precios de apertura - 100 últimas barras será suficiente
   int copied=CopyHigh(Symbol(),0,0,100,high);
   if(copied<=0)
     {
      Print("Fallo al copiar los valores de la serie de precios High");
      return;
     }
//--- vamos a copiar los precios de cierre - 100 últimas barras será suficiente
   copied=CopyLow(Symbol(),0,0,100,low);
   if(copied<=0)
     {
      Print("Fallo al copiar los valores de la serie de precios Low");
      return;
     }
//--- vamos a copiar la hora de apertura para 100 últimas barras
   copied=CopyTime(Symbol(),0,0,100,time);
   if(copied<=0)
     {
      Print("Fallo al copiar los valores de la serie de precios Time");
      return;
     }
//--- vamos a organizar el acceso a los datos copiados como en las series temporales -
   ArraySetAsSeries(high,true);
   ArraySetAsSeries(low,true);
   ArraySetAsSeries(time,true);

//--- coordinadas del primer punto de anclaje del objeto Fibo
   price1=high[70];
   time1=time[70];
//--- coordinadas del segundo punto de anclaje del objeto Fibo
```

© 2000-2025, MetaQuotes Ltd.

```
   price2=low[50];
   time2=time[50];

//--- ya es hora de crear el mismo objeto Fibo
   bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
   if(created) // si el objeto ha sido creado con éxito
     {
      //--- vamos a configurar los colores de niveles Fibo
      ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
      //--- por cierto, ¿cuántos niveles Fibo tenemos nosotros
      int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
      Print("Fibo levels before = ",levels);
      //---mostramos en el Diario => número del nivel:valor de descripción_de_nivel
      for(int i=0;i<levels;i++)
        {
         Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
               "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
        }
      //--- vamos a intentar aumentar la cantidad de niveles a uno
      bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
      if(!modified) // no ha salido cambiar la cantidad de niveles
        {
         Print("Fallo al cambiar el número de niveles de Fibo, error ",GetLastError())
        }
      //--- simplemente avisamos
      Print("Fibo levels after =",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS));
      //--- establecemos el valor para el nivel recién creado
      bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
      if(added) // hemos podido establecer el valor para este nivel
        {
         Print("Hemos establecido con éxito otro nivel más de Fibo");
         //--- también hay que establecer la descripción del nivel
         ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
         ChartRedraw(0);
         //--- obtenemos el valor actual de la cantidad de niveles en el objeto Fibo
         levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
         Print("Fibo levels after adding = ",levels);
         //--- otra vez visualizamos todos los niveles, simplemente para comprobar
         for(int i=0;i<levels;i++)
           {
            Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                  "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
           }

        }
      else // fallo al intentar aumentar la cantidad de niveles en el objeto Fibo
        {
         Print("Fallo al establecer otro nivel más de Fibo. Error",GetLastError());
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
```

Véase también

Tipos de objetos, Propiedades de objetos

© 2000-2025, MetaQuotes Ltd.

ObjectSetInteger

Establece el valor de la propiedad correspondiente de un objeto. La propiedad del objeto debe ser de los tipos datetime, int, color, bool o char. Existen 2 variantes de la función.

Configuración del valor de la propiedad sin modificador

```
bool  ObjectSetInteger(
   long                              chart_id,          // identificador del gráfico
   string                            name,              // nombre
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // propiedad
   long                              prop_value         // valor
   );
```

Configuración del valor de la propiedad indicando el modificador

```
bool  ObjectSetInteger(
   long                              chart_id,          // identificador del gráfico
   string                            name,              // nombre
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // propiedad
   int                               prop_modifier,     // modificador
   long                              prop_value         // valor
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto.

prop_id

[in] Identificador de la propiedad del objeto. Su valor puede ser uno de los valores de la enumeración ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Modificador de la propiedad especificada. La mayoría de las propiedades no requiere el modificador.

prop_value

[in] Valor de la propiedad.

Valor devuelto

Devuelve true sólo si el comando de modificar las propiedades de un objeto gráfico se ha enviado con éxito al gráfico. De lo contrario devuelve false. Para la información más detallada sobre el error hay que llamar a la función GetLastError().

Nota

© 2000-2025, MetaQuotes Ltd.

La función usa una llamada asincrónica, esto significa que la función no espera a la ejecución del comando que se ha colocado con éxito en la cola del gráfico indicado, sino que devuelve el control de inmediato.

Para comprobar el resultado de la ejecución en un gráfico ajeno se puede usar una función que solicite la propiedad indicada del objeto. Pero, en esta caso, además, se deberá tener en cuenta que estas funciones se colocan al final de la cola de comandos del gráfico y esperan el resultado de la ejecución, es decir, pueden consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

Un ejemplo de cómo crear una tabla de colores Web

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                Table of Web Colors|
//|                        Copyright 2011, MetaQuotes Software Corp |
//|                                             https://www.metaquotes.net |
//+------------------------------------------------------------------+
#define X_SIZE 140      // ancho del objeto OBJ_EDIT
#define Y_SIZE 33       // ancho del objeto OBJ_EDIT
//+------------------------------------------------------------------+
//| Array de colores Web                                             |
//+------------------------------------------------------------------+
color ExtClr[140]=
  {
   clrAliceBlue,clrAntiqueWhite,clrAqua,clrAquamarine,clrAzure,clrBeige,clrBisque,clrB
   clrBlue,clrBlueViolet,clrBrown,clrBurlyWood,clrCadetBlue,clrChartreuse,clrChocolate
   clrCornsilk,clrCrimson,clrCyan,clrDarkBlue,clrDarkCyan,clrDarkGoldenrod,clrDarkGray
   clrDarkMagenta,clrDarkOliveGreen,clrDarkOrange,clrDarkOrchid,clrDarkRed,clrDarkSalm
   clrDarkSlateBlue,clrDarkSlateGray,clrDarkTurquoise,clrDarkViolet,clrDeepPink,clrDee
   clrDodgerBlue,clrFireBrick,clrFloralWhite,clrForestGreen,clrFuchsia,clrGainsboro,cl
   clrGoldenrod,clrGray,clrGreen,clrGreenYellow,clrHoneydew,clrHotPink,clrIndianRed,cl
   clrLavender,clrLavenderBlush,clrLawnGreen,clrLemonChiffon,clrLightBlue,clrLightCora
   clrLightGoldenrod,clrLightGreen,clrLightGray,clrLightPink,clrLightSalmon,clrLightSe
   clrLightSlateGray,clrLightSteelBlue,clrLightYellow,clrLime,clrLimeGreen,clrLinen,cl
   clrMediumAquamarine,clrMediumBlue,clrMediumOrchid,clrMediumPurple,clrMediumSeaGreen
   clrMediumSpringGreen,clrMediumTurquoise,clrMediumVioletRed,clrMidnightBlue,clrMintC
   clrNavajoWhite,clrNavy,clrOldLace,clrOlive,clrOliveDrab,clrOrange,clrOrangeRed,clrO
   clrPaleGreen,clrPaleTurquoise,clrPaleVioletRed,clrPapayaWhip,clrPeachPuff,clrPeru,c
   clrPurple,clrRed,clrRosyBrown,clrRoyalBlue,clrSaddleBrown,clrSalmon,clrSandyBrown,c
   clrSienna,clrSilver,clrSkyBlue,clrSlateBlue,clrSlateGray,clrSnow,clrSpringGreen,clr
   clrThistle,clrTomato,clrTurquoise,clrViolet,clrWheat,clrWhite,clrWhiteSmoke,clrYell
  };
//+------------------------------------------------------------------+
//| Creación e inicialización del objeto OBJ_EDIT                    |
//+------------------------------------------------------------------+
void CreateColorBox(int x,int y,color c)
  {
//--- generaremos el nombre para nuevo objeto según el nombre del color
   string name="ColorBox_"+(string)x+"_"+(string)y;
//--- crearemos nuevo objeto OBJ_EDIT
   if(!ObjectCreate(0,name,OBJ_EDIT,0,0,0))
     {
      Print("Cannot create: '",name,"'");
      return;
     }
//--- fijaremos las coordinadas del punto de enlace, ancho y alto en píxeles
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x*X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y*Y_SIZE);
   ObjectSetInteger(0,name,OBJPROP_XSIZE,X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YSIZE,Y_SIZE);
//--- estableceremos el color del texto del objeto
   if(clrBlack==c) ObjectSetInteger(0,name,OBJPROP_COLOR,clrWhite);
   else            ObjectSetInteger(0,name,OBJPROP_COLOR,clrBlack);
//--- estableceremos el color del fondo
   ObjectSetInteger(0,name,OBJPROP_BGCOLOR,c);
//--- estableceremos el texto del objeto OBJ_EDIT correspondiente al color del fondo
   ObjectSetString(0,name,OBJPROP_TEXT,(string)c);
  }
//+------------------------------------------------------------------+
//| Función del inicio del script                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- crearemos una tabla de los bloques coloreados 7x20
   for(uint i=0;i<140;i++)
      CreateColorBox(i%7,i/7,ExtClr[i]);
  }
```

Véase también

Tipos de objetos, Propiedades de objetos

© 2000-2025, MetaQuotes Ltd.

ObjectSetString

Establece el valor de la propiedad correspondiente de un objeto. La propiedad del objeto debe ser del tipo string. Existen 2 variantes de la función.

Configuración del valor de la propiedad sin modificador

```
bool  ObjectSetString(
   long                          chart_id,          // identificador del gráfico
   string                        name,              // nombre
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // propiedad
   string                        prop_value         // valor
   );
```

Configuración del valor de la propiedad indicando el modificador

```
bool  ObjectSetString(
   long                          chart_id,          // identificador del gráfico
   string                        name,              // nombre
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // propiedad
   int                           prop_modifier,     // modificador
   string                        prop_value         // valor
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto.

prop_id

[in] Identificador de la propiedad del objeto. Su valor puede ser uno de los valores de la enumeración ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Modificador de la propiedad especificada. La mayoría de las propiedades no requiere el modificador. Significa el número del nivel en las herramientas de Fibonacci y en el objeto gráfico Tridente de Andrews. La numeración se empieza desde cero.

prop_value

[in] Valor de la propiedad.

Valor devuelto

Devuelve true sólo si el comando de modificar las propiedades de un objeto gráfico se ha enviado con éxito al gráfico. De lo contrario devuelve false. Para la información más detallada sobre el error hay que llamar a la función GetLastError().

Nota

© 2000-2025, MetaQuotes Ltd.

La función usa una llamada asincrónica, esto significa que la función no espera a la ejecución del comando que se ha colocado con éxito en la cola del gráfico indicado, sino que devuelve el control de inmediato.

Para comprobar el resultado de la ejecución en un gráfico ajeno se puede usar una función que solicite la propiedad indicada del objeto. Pero, en esta caso, además, se deberá tener en cuenta que estas funciones se colocan al final de la cola de comandos del gráfico y esperan el resultado de la ejecución, es decir, pueden consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

Al renombrar un objeto gráfico, se generan al mismo tiempo dos eventos que pueden ser procesados en el Asesor Experto o el indicador usando la función OnChartEvent():

- evento de eliminación del objeto con el nombre anterior;
- evento de creación del objeto gráfico con el nombre nuevo.

© 2000-2025, MetaQuotes Ltd.

ObjectGetDouble

Devuelve el valor de la correspondiente propiedad del objeto. La propiedad del objeto debe ser del tipo double. Existen 2 variantes de la función.

1 Devuelve el valor de la propiedad directamente.

```
double  ObjectGetDouble(
   long                          chart_id,          // identificador del gráfico
   string                        name,              // nombre del objeto
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // identificador de la propiedad
   int                           prop_modifier=0    // modificador de la propiedad,
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso de éxito el valor de la propiedad se coloca en una variable receptora que se pasa por referencia por el último parámetro.

```
bool  ObjectGetDouble(
   long                          chart_id,          // identificador del gráfico
   string                        name,              // nombre del objeto
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // identificador de la propiedad
   int                           prop_modifier,     // modificador de la propiedad
   double&                       double_var         // aquí recibimos el valor de la
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto.

prop_id

[in] Identificador de la propiedad del objeto. Su valor puede ser uno de los valores de la enumeración ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Modificador de la propiedad especificada. Para la primera opción el valor del modificador es igual a 0 por defecto. La mayoría de las propiedades no requiere el modificador. Significa el número del nivel en las herramientas de Fibonacci y en el objeto gráfico Tridente de Andrews. La numeración se empieza desde cero.

double_var

[out] Variable del tipo double que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo double para la primera opción de la llamada.

Para la segunda variante devuelve true, si dicha propiedad se mantiene y su valor ha sido colocado en la variable double_var, de lo contrario devuelve false. Para la información más detallada sobre el error hay que llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

Nota

La función usa una llamada sincrónica, esto significa que la función espera a la ejecución de todos los comandos que han sido ubicados en la cola del gráfico antes de su llamada, y por eso puede consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

© 2000-2025, MetaQuotes Ltd.

ObjectGetInteger

Devuelve el valor de la correspondiente propiedad del objeto. La propiedad del objeto debe ser de los tipos datetime, int, color, bool o char. Existen 2 variantes de la función.

1 Devuelve el valor de la propiedad directamente.

```
long  ObjectGetInteger(
   long                          chart_id,          // identificador del gráfico
   string                        name,              // nombre del objeto
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // identificador de la propieda
   int                           prop_modifier=0    // modificador de la propiedad,
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso de éxito el valor de la propiedad se coloca en una variable receptora que se pasa por referencia por el último parámetro.

```
bool  ObjectGetInteger(
   long                          chart_id,          // identificador del gráfico
   string                        name,              // nombre del objeto
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // identificador de la propieda
   int                           prop_modifier,     // modificador de la propiedad
   long&                         long_var           // aquí recibimos el valor de l
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto.

prop_id

[in] Identificador de la propiedad del objeto. Su valor puede ser uno de los valores de la enumeración ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Modificador de la propiedad especificada. Para la primera opción el valor del modificador es igual a 0 por defecto. La mayoría de las propiedades no requiere el modificador. Significa el número del nivel en las herramientas de Fibonacci y en el objeto gráfico Tridente de Andrews. La numeración se empieza desde cero.

long_var

[out] Variable del tipo long que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo long para la primera opción de la llamada.

Para la segunda variante devuelve true, si dicha propiedad se mantiene y su valor ha sido colocado en la variable long_var, de lo contrario devuelve false. Para la información más detallada sobre el error hay que llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

Nota

La función usa una llamada sincrónica, esto significa que la función espera a la ejecución de todos los comandos que han sido ubicados en la cola del gráfico antes de su llamada, y por eso puede consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

© 2000-2025, MetaQuotes Ltd.

ObjectGetString

Devuelve el valor de la correspondiente propiedad del objeto. La propiedad del objeto debe ser del tipo string. Existen 2 variantes de la función.

1 Devuelve el valor de la propiedad directamente.

```
string  ObjectGetString(
   long                          chart_id,          // identificador del gráfico
   string                        name,              // nombre del objeto
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // identificador de la propiedad
   int                           prop_modifier=0    // modificador de la propiedad,
   );
```

2. Devuelve true o false dependiendo del éxito de ejecución de la función. En caso de éxito el valor de la propiedad se coloca en una variable receptora que se pasa por referencia por el último parámetro.

```
bool  ObjectGetString(
   long                          chart_id,          // identificador del gráfico
   string                        name,              // nombre del objeto
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // identificador de la propiedad
   int                           prop_modifier,     // modificador de la propiedad
   string&                       string_var         // aquí recibimos el valor de la
   );
```

Parámetros

chart_id

[in] Identificador del gráfico. 0 significa el gráfico actual.

name

[in] Nombre del objeto.

prop_id

[in] Identificador de la propiedad del objeto. Su valor puede ser uno de los valores de la enumeración ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Modificador de la propiedad especificada. Para la primera opción el valor del modificador es igual a 0 por defecto. La mayoría de las propiedades no requiere el modificador. Significa el número del nivel en las herramientas de Fibonacci y en el objeto gráfico Tridente de Andrews. La numeración se empieza desde cero.

string_var

[out] Variable del tipo string que recibe el valor de la propiedad requerida.

Valor devuelto

Valor del tipo string para la primera opción de la llamada.

Para la segunda variante devuelve true, si dicha propiedad se mantiene y su valor ha sido colocado en la variable string_var, de lo contrario devuelve false. Para la información más detallada sobre el error hay que llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

Nota

La función usa una llamada sincrónica, esto significa que la función espera a la ejecución de todos los comandos que han sido ubicados en la cola del gráfico antes de su llamada, y por eso puede consumir un tiempo considerable. Hay que tener esta circunstancia en cuenta al trabajar con multitud de objetos en el gráfico.

Al renombrar un objeto gráfico, se generan al mismo tiempo dos eventos que pueden ser procesados en el Asesor Experto o el indicador usando la función OnChartEvent():

- evento de eliminación del objeto con el nombre anterior;
- evento de creación del objeto gráfico con el nombre nuevo.

© 2000-2025, MetaQuotes Ltd.

TextSetFont

Establece la fuente para visualizar el texto a través de los métodos del dibujo y devuelve el resultado de esta operación. Por defecto se usa la fuente Arial con el tamaño -120 (12 pt).

```
bool  TextSetFont(
   const string  name,            // nombre de la fuente o la ruta del archivo de la f
   int           size,            // tamaño de la fuente
   uint          flags,           // combinación de banderas
   int           orientation=0    // ángulo de inclinación del texto
   );
```

Parámetros

name

[in] Nombre de la fuente en el sistema, o el nombre del recurso que contiene la fuente, o la ruta del archivo de la fuente en el disco.

size

[in] Tamaño de la fuente que puede ser establecido con valores positivos y negativos. En caso de valores positivos, el tamaño del texto a mostrar no depende de los ajustes de tamaños de las fuentes en el sistema operativo. En caso de valores negativos, el valor se establece en las décimas partes del punto y el tamaño del texto va a depender de los ajustes del sistema ("escala estándar" o "escala grande"). Más detalles sobre la diferencia entre los regímenes se puede leer en la Nota.

flags

[in] Combinación de banderas que describen el estilo de la fuente.

orientation

[in] Ángulo de inclinación del texto por la horizontal respecto al eje X, la unidad de medición es de 0,1 del grado. Es decir, *orientation=450* significa la inclinación bajo un ángulo de 45 grados.

Valor devuelto

Devuelve true si la fuente actual se establece con éxito, de lo contrario es false. Posibles códigos de errores:

- ERR_INVALID_PARAMETER(4003) - *name* representa NULL o "" (cadena vacía),
- ERR_INTERNAL_ERROR(4001) - error del sistema operativo (por ejemplo, intento de crear una fuente que no existe).

Nota

Si en el nombre de la fuente se utiliza "::", la fuente se carga desde un recurso EX5. Si el nombre de la fuente *name* va acompañado con la extensión, entonces esta fuente se carga desde el archivo. En esta caso, si la ruta de la fuente se empieza con "\" o "/", el archivo se busca referente la carpeta MQL5, de lo contrario se busca referente a la ruta del archivo EX5 que ha llamado a la función TextSetFont().

El tamaño de la fuente se establece con valores positivos o negativos, el signo determina la dependencia del tamaño de la fuente de los ajustes del sistema operativo (la escala de la fuente).

- Si para establecer el tamaño se utiliza un número positivo, este tamaño se transforma en unidades físicas de medición del dispositivo (píxeles) durante el cambio de la fuente lógica a la

© 2000-2025, MetaQuotes Ltd.

física, y este tamaño corresponde a la altura de los glifos del símbolo escogidos desde las fuentes disponibles. Este caso no es recomendable cuando se supone el uso común de los textos visualizados por la función TextOut() y los textos visualizados a través del objeto gráfico OBJ_LABEL ("Etiqueta de texto") en el mismo gráfico.

- Si para establecer el tamaño se utiliza un número negativo, se supone que el número especificado se establece en décimas partes del punto lógico (el valor -350 es igual a 35 puntos lógicos) y se divide por 10. Luego el valor obtenido se transforma en unidades físicas de medición del dispositivo (píxeles) y corresponde al valor absoluto de la altura del símbolo desde las fuentes disponibles. Para conseguir en la pantalla el texto con el mismo tamaño que hay en el objeto OBJ_LABEL, es necesario multiplicar el tamaño de la fuente especificada en las propiedades del objeto por -10.

Las banderas pueden ser utilizadas como combinaciones de las banderas del estilo con una de las banderas que determina el grosor de la fuente. Los nombres de las banderas se listan a continuación.

Banderas para fijar el estilo de escritura de la fuente

Bandera	Descripción
FONT_ITALIC	Cursiva
FONT_UNDERLINE	Subrayado
FONT_STRIKEOUT	Tachado

Banderas para fijar el grosor de la fuente

Bandera
FW_DONTCARE
FW_THIN
FW_EXTRALIGHT
FW_ULTRALIGHT
FW_LIGHT
FW_NORMAL
FW_REGULAR
FW_MEDIUM
FW_SEMIBOLD
FW_DEMIBOLD
FW_BOLD
FW_EXTRABOLD
FW_ULTRABOLD
FW_HEAVY
FW_BLACK

© 2000-2025, MetaQuotes Ltd.

Véase también

Recursos, ResourceCreate(), ResourceSave(), TextOut()

© 2000-2025, MetaQuotes Ltd.

TextOut

Esta función visualiza el texto en un array personalizado (búfer) y devuelve el resultado de esta operación. Este array está destinado para la generación de un recurso gráfico.

```
bool  TextOut(
   const string       text,          // texto mostrado
   int                x,             // coordenada X
   int                y,             // coordenada Y
   uint               anchor,        // modo de anclaje
   uint               &data[],       // búfer de salida
   uint               width,         // ancho del búfer en píxeles
   uint               height,        // alto del búfer en píxeles
   uint               color,         // color del texto
   ENUM_COLOR_FORMAT  color_format   // formato del texto a visualizar
   );
```

Parámetros

text

[in] Texto visualizado que va a ser escrito en el búfer. Se visualiza sólo el texto de una solo línea.

x

[in] Coordenada X del punto de anclaje para el texto visualizado.

y

[in] Coordenada Y del punto de anclaje para el texto visualizado.

anchor

[in] Valor del conjunto de 9 modos predeterminados de la posición del punto de anclaje del texto visualizado. La posición se establece con la combinación de dos banderas: la bandera de alineación del texto por la horizontal y la bandera de alineación del texto por la vertical. Los nombres de las banderas se listan en la Nota de abajo.

data[]

[in] Búfer que recibe el texto. Este búfer se utiliza para crear un recurso gráfico.

width

[in] Ancho del búfer en puntos (píxeles).

height

[in] Alto del búfer en puntos (píxeles).

color

[in] Color del texto.

color_format

[in] Formato del color se establece con un valor desde la enumeración ENUM_COLOR_FORMAT.

Valor devuelto

Devuelve true en caso de la ejecución exitosa, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

Nota

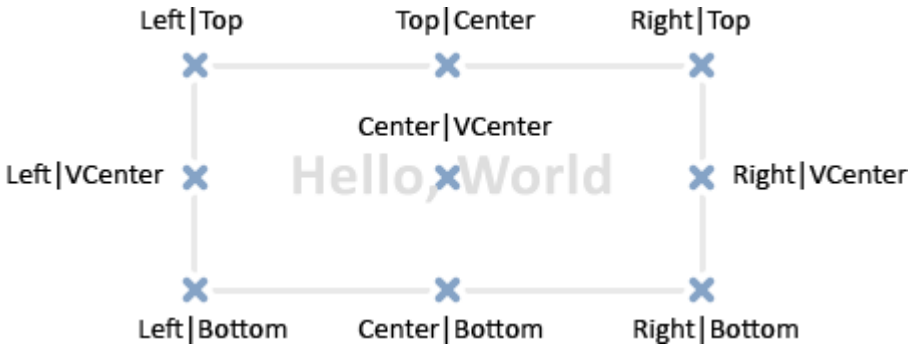
El modo de anclaje que se establece con el parámetro `anchor` es una combinación de dos banderas de alineación de texto por la vertical y por la horizontal. Banderas de alineación de texto por la horizontal:

- TA_LEFT - punto de anclaje situado en la parte izquierda del rectángulo limitador
- TA_CENTER - punto de anclaje por la horizontal se sitúa en el centro del rectángulo limitador
- TA_RIGHT - punto de anclaje situado en la parte derecha del rectángulo limitador

Banderas de alineación de texto por la vertical

- TA_TOP - punto de anclaje situado en la parte superior del rectángulo limitador
- TA_VCENTER - punto de anclaje por la vertical se sitúa en el centro del rectángulo limitador
- TA_BOTTOM - punto de anclaje situado en la parte inferior del rectángulo limitador

Las posibles combinaciones de banderas y los modos de anclaje que éstas establecen se muestran en la figura de abajo.



Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//--- ancho y alto del canvas (lienzo en el que se hace el dibujo)
#define IMG_WIDTH  200
#define IMG_HEIGHT 200
//--- antes de iniciar el script mostramos la ventana con los parámetros
#property script_show_inputs
//--- damos la posibilidad de establecer el formato del color
input ENUM_COLOR_FORMAT clr_format=COLOR_FORMAT_XRGB_NOALPHA;
//--- array (búfer) de dibujar
uint ExtImg[IMG_WIDTH*IMG_HEIGHT];
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- creamos el objeto OBJ_BITMAP_LABEL para dibujar
   ObjectCreate(0,"CLOCK",OBJ_BITMAP_LABEL,0,0,0);
//--- especificamos el nombre del recurso gráfico para escribir en el objeto CLOCK
   ObjectSetString(0,"CLOCK",OBJPROP_BMPFILE,"::IMG");

//--- variables auxiliares
   double a;            // ángulo de la flecha
   uint   nm=2700;      // contador de minutos
   uint   nh=2700*12;   // contador de horas
   uint   w,h;          // variables para obtener los tamaños de las cadenas de texto
   int    x,y;          // variables para calcular las coordenadas actuales del punto

//--- rotamos las agujas del reloj en un ciclo infinito hasta la detención del script
   while(!IsStopped())
     {
      //--- limpiamos el array del búfer para el dibujo del reloj
      ArrayFill(ExtImg,0,IMG_WIDTH*IMG_HEIGHT,0);
      //--- establecer la fuente para dibujar las cifras sobre la carátula del reloj
      TextSetFont("Courier",30,FW_EXTRABOLD,0);
      //--- dibujamos la carátula del reloj
      for(int i=1;i<=12;i++)
        {
         //--- obtenemos el tamaño de la hora actual sobre la carátula
         TextGetSize(string(i),w,h);
         //--- calculamos coordenadas de la hora actual sobre la carátula
         a=-((i*300)%3600*M_PI)/1800.0;
         x=IMG_WIDTH/2-int(sin(a)*80+0.5+w/2);
         y=IMG_HEIGHT/2-int(cos(a)*80+0.5+h/2);
         //--- visualización de esta hora sobre la carátula en el búfer ExtImg[]
         TextOut(string(i),x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,c
        }
      //--- ahora establecemos la fuente para dibujar el minutero
      TextSetFont("Courier",20,FW_EXTRABOLD,(uint)-(nm%3600));
      //--- obtenemos el tamaño del minutero
      TextGetSize("---->",w,h);
      //--- calculamos las coordenadas del minutero sobre la carátula del reloj
      a=-(nm%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- visualización del minutero sobre la carátula en el búfer ExtImg[]
      TextOut("---->",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_fo

      //--- ahora establecemos la fuente para dibujar el horario
      TextSetFont("Courier",20,FW_EXTRABOLD,(uint)-(nh/12%3600));
      TextGetSize("===>",w,h);
      //--- calculamos las coordenadas del horario sobre la carátula del reloj
      a=-(nh/12%3600*M_PI)/1800.0;
```

© 2000-2025, MetaQuotes Ltd.

```
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- visualización del horario sobre la carátula en el búfer ExtImg[]
      TextOut("===>",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_fo

      //--- actualización del recurso gráfico
      ResourceCreate("::IMG",ExtImg,IMG_WIDTH,IMG_HEIGHT,0,0,IMG_WIDTH,clr_format);
      //--- actualización forzada del gráfico
      ChartRedraw();

      //--- aumentamos los contadores de hora y minutos
      nm+=60;
      nh+=60;
      //--- mantenemos una pausa entre los cuadros
      Sleep(100);
     }
//--- eliminamos el objeto CLOCK al terminarse el trabajo del script
   ObjectDelete(0,"CLOCK");
//---
  }
```

Véase también

Recursos, ResourceCreate(), ResourceSave(), TextGetSize(), TextSetFont()

© 2000-2025, MetaQuotes Ltd.

TextGetSize

Devuelve el alto y el ancho de la cadena con actuales ajustes de la fuente.

```
bool  TextGetSize(
   const string       text,          // cadena del texto
   uint&               width,        // ancho del búfer en píxeles
   uint&               height        // alto del búfer en píxeles
   );
```

Parámetros

text

[in] Cadena para la que obtenemos el largo y el ancho.

width

[out] Parámetro de entrada para obtener el ancho.

height

[out] Parámetro de entrada para obtener el alto.

Valor devuelto

Devuelve true en caso de la ejecución exitosa, de lo contrario devuelve false. Posibles códigos de errores:

- ERR_INTERNAL_ERROR(4001) - en caso del error del sistema operativo.

Véase también

Recursos, ResourceCreate(), ResourceSave(), TextSetFont(), TextOut()

© 2000-2025, MetaQuotes Ltd.

Funciones para trabajar con indicadores técnicos

Todas las funciones del tipo iMA, iAC, iMACD, iIchimoku etc. crean una copia del indicador técnico correspondiente en la caché global del terminal de cliente. Si la copia del indicador con estos parámetros ya existe, una copia nueva no se crea, sólo el contador de referencias a esta copia existente se aumenta.

Estas funciones devuelven el manejador (handle) de la copia del indicador correspondiente. En el futuro, si usamos este manejador, se puede recibir los datos calculados por el indicador correspondiente. Los datos del buffer correspondiente (los indicadores técnicos contienen los datos calculados en sus buffers internos el número de los cuales puede ser de 1 a 5, dependiendo de cada indicador) pueden ser copiados en un programa mql5 usando la función CopyBuffer().

No se puede acudir a los datos del indicador justo después de su creación porque el cálculo de valores de indicador requiere un tiempo, por eso es mejor crear los manejadores de indicadores en OnInit(). La función iCustom() crea el indicador personalizado correspondiente y devuelve su manejador en caso de crearlo con éxito. Los indicadores personalizados pueden contener de hasta 512 buffers de indicador, el contenido de los cuales también se puede obtener mediante la función CopyBuffer(), usando el manejador obtenido.

Existe un método universal para crear cualquier indicador técnico con la función IndicatorCreate(). Esta función acepta los siguientes parámetros de entrada:

- nombre del símbolo;
- período de tiempo;
- tipo del indicador que se crea;
- número de parámetros de entrada del indicador;
- array del tipo MqlParam que contiene todos los parámetros de entrada necesarios.

Para liberar la memoria del ordenador de un indicador que ya no se usa existe la función IndicatorRelease() a la que se pasa el manejador de este indicador.

Nota. Llamada repetida a la función del indicador con los mismos parámetros dentro de un programa mql5 no llevará al aumento repetido del contador de referencias; el contador será aumentado sólo una vez a 1. Sin embargo, se recomienda obtener los manejadores de indicadores en la función OnInit() o en el constructor de clase, con su posterior uso en las demás funciones. El contador de referencias se disminuye cuando un programa mql5 se deinicializa.

Todas las funciones de indicador disponen como mínimo de 2 parámetros: símbolo y período. El valor del símbolo NULL significa el instrumento actual, el valor del período 0 significa el período de tiempo corriente.

Función	Devuelve el manejador del indicador:
iAC	Accelerator Oscillator
iAD	Accumulation/Distribution
iADX	Average Directional Index
iADXWilder	Average Directional Index by Welles Wilder
iAlligator	Alligator

© 2000-2025, MetaQuotes Ltd.

Función	Devuelve el manejador del indicador:
iAMA	Adaptive Moving Average
iAO	Awesome Oscillator
iATR	Average True Range
iBearsPower	Bears Power
iBands	Bollinger Bands®
iBullsPower	Bulls Power
iCCI	Commodity Channel Index
iChaikin	Chaikin Oscillator
iCustom	indicador personalizado
iDEMA	Double Exponential Moving Average
iDeMarker	DeMarker
iEnvelopes	Envelopes
iForce	Force Index
iFractals	Fractals
iFrAMA	Fractal Adaptive Moving Average
iGator	Gator Oscillator
iIchimoku	Ichimoku Kinko Hyo
iBWMFI	Market Facilitation Index by Bill Williams
iMomentum	Momentum
iMFI	Money Flow Index
iMA	Moving Average
iOsMA	Moving Average of Oscillator (MACD histogram)
iMACD	Moving Averages Convergence-Divergence
iOBV	On Balance Volume
iSAR	Parabolic Stop And Reverse System
iRSI	Relative Strength Index
iRVI	Relative Vigor Index
iStdDev	Standard Deviation
iStochastic	Stochastic Oscillator
iTEMA	Triple Exponential Moving Average

© 2000-2025, MetaQuotes Ltd.

Función	Devuelve el manejador del indicador:
iTriX	Triple Exponential Moving Averages Oscillator
iWPR	Williams' Percent Range
iVIDyA	Variable Index Dynamic Average
iVolumes	Volumes

© 2000-2025, MetaQuotes Ltd.

iAC

La función crea el indicador Accelerator Oscillator en la caché global del terminal de cliente y devuelve su manejador. Tiene sólo un búfer.

```
int  iAC(
   string           symbol,     // nombre del símbolo
   ENUM_TIMEFRAMES  period      // período
   );
```

Parámetros

symbol

[in] Símbolo de un instrumento financiero, cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso de fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador de un indicador que ya no se usa, se usa la función IndicatorRelease() a la que se le pasa el manejador de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAC.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iAC."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- Construcción de iAC
#property indicator_label1  "iAC"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAC,               // usar iAC
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iAC;          // tipo de función
input string                symbol=" ";             // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // período de tiempo
//--- búfers indicadores
double iACBuffer[];
double iACColors[];
//--- variable para guardar el manejador del indicador iAC
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Accelerator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iAC)
      handle=iAC(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AC);
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- avisaremos sobre el fallo y mostraremos el número del error
       PrintFormat("Fallo al crear el manejador del indicador iAC para el par %s/%s, có
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- el trabajo del indicador se finaliza anticipadamente
       return(INIT_FAILED);
      }
//--- mostraremos para qué par símbolo/período ha sido calculado el indicador Accelera
   short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iAC
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iACBuffer es más grande que los números de valores en el indic
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- significa que no es la primera vez que se calcula nuestro indicador y desc
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays iACBuffer y iACColors con los valores desde el indicador Acc
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Accelerator Oscillator
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iAC           |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // búfer indicador de valores Accel
                          double &color_indexes[], // búfer de color (para almacenar l
                          int ind_handle,          // manejador del indicador iAC
                          int amount               // número de valores a copiar
                          )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iACBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iAC, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- ahora copiaremos los índices de colores
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- si el copiado ha fallado, mostramos el código del error
      PrintFormat("Fallo al copiar los valores de los colores desde el indicador iAC,
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAD

Devuelve el manejador del indicador Accumulation/Distribution. Tiene sólo un búfer.

```
int  iAD(
   string               symbol,             // nombre del símbolo
   ENUM_TIMEFRAMES      period,             // período
   ENUM_APPLIED_VOLUME  applied_volume      // tipo de volumen para el cálculo
   );
```

Parámetros

symbol

[in] Símbolo de un instrumento financiero, cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

applied_volume

[in] Volumen usado. Puede ser cualquier valor de la enumeración ENUM_APPLIED_VOLUME.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                    Demo_iAD.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iAD."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iAD
#property indicator_label1  "iAD"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAD,               // usar iAD
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iAD;          // tipo de función
input ENUM_APPLIED_VOLUME   volumes;                // volumen que se utiliza
input string                symbol=" ";             // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // período de tiempo
//--- búfer indicador
double          iADBuffer[];
//--- variable para guardar el manejador del indicador iAD
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Accumulation/Distributi
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iAD)
      handle=iAD(name,period,volumes);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=volumes;
      handle=IndicatorCreate(name,period,IND_AD,1,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iAD para el par %s/%s, có
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos para qué par símbolo/período ha sido calculado el indicador Accumula
   short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iAD
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- si el array iADBuffer supera el número de valores en el indicador iAD para
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iADBuffer con los valores desde el indicador Accumulation/Dist
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Accumulation/Distribution
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iAD           |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // búfer indicador de la línea Accumulati
                         int ind_handle,     // manejador del indicador iAD
                         int amount          // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- lenamos la parte del array iADBuffer con los valores desde el búfer indicador ba
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iAD, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iADX

Devuelve el manejador del indicador Average Directional Movement Index.

```
int  iADX(
   string           symbol,          // nombre del símbolo
   ENUM_TIMEFRAMES  period,          // período
   int              adx_period       // período promedio
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

adx_period

[in] Período de cálculo del índice.

Nota

Números de búfers: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iADX.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iADX."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

© 2000-2025, MetaQuotes Ltd.

```
//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADX,              // usar iADX
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iADX;         // tipo de función
input int                  adx_period=14;          // período de cálculo
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // período de tiempo
//--- búfers indicadores
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- variable para guardar el manejador del indicador iADX
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Average Directional Mov
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de arrays a los búfers indicadores
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iADX)
      handle=iADX(name,period,adx_period);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADX,1,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iADX para el par %s/%s, c
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos para qué par símbolo/período ha sido calculado el indicador Average
   short_name=StringFormat("iADX(%s/%s period=%d)",name,EnumToString(period),adx_perio
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iADX
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array ADXBuffer supera el número de valores en el indicador iADX pa
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array con los valores desde el indicador Average Directional Movemen
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Average Directional Movement I
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iADX          |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool FillArraysFromBuffers(double &adx_values[],      // búfer indicador de la línea A
                           double &DIplus_values[],   // búfer indicador para DI+
                           double &DIminus_values[],  // búfer indicador para DI-
                           int ind_handle,            // manejador del indicador iADXW
                           int amount                 // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array ADXBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iADX, código del erro
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array DI_plusBuffer con los valores desde el búfer indica
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iADX, código del erro
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array DI_plusBuffer con los valores desde el búfer indica
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iADX, código del erro
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iADXWilder

Devuelve el manejador del indicador Average Directional Movement Index by Welles Wilder.

```
int  iADXWilder(
   string           symbol,          // nombre del símbolo
   ENUM_TIMEFRAMES  period,          // período
   int              adx_period       // período promedio
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

adx_period

[in] Período de cálculo del índice.

Nota

Números de búfers: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                  iADXWilder.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iADXWilder."
#property description "El símbolo y el período de tiempo en el que se calcula el indic"
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

© 2000-2025, MetaQuotes Ltd.

```
//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADXWilder,         // usar iADXWilder
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iADXWilder;   // tipo de función
input int                  adx_period=14;          // período de cálculo
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // período de tiempo
//--- búfers indicadores
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- variable para guardar el manejador del indicador iADXWilder
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Average Directional Mov
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de arrays a los búfers indicadores
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iADXWilder)
      handle=iADXWilder(name,period,adx_period);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADXW,1,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iADXWilder para el par %s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos para qué par símbolo/período se calcula el indicador Average Directi
   short_name=StringFormat("iADXWilder(%s/%s period=%d)",name,EnumToString(period),adx
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iADXWilder
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array ADXBuffer supera el número de valores en el indicador iADXWild
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array con los valores desde el indicador Average Directional Movemen
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Average Directional Movement I
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iADXWilder    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool FillArraysFromBuffers(double &adx_values[],      // búfer indicador de la línea A
                           double &DIplus_values[],   // búfer indicador para DI+
                           double &DIminus_values[],  // búfer indicador para DI-
                           int ind_handle,            // manejador del indicador iADXW
                           int amount                 // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array ADXBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iADXWilder, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array DI_plusBuffer con los valores desde el búfer indica
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iADXWilder, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array DI_plusBuffer con los valores desde el búfer indica
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iADXWilder, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAlligator

Devuelve el manejador del indicador Alligator.

```
int  iAlligator(
   string               symbol,           // nombre del símbolo
   ENUM_TIMEFRAMES      period,           // período
   int                  jaw_period,       // período para el cálculo de mandíbulas
   int                  jaw_shift,        // desplazamiento horizontal de mandíbulas
   int                  teeth_period,     // período para el cálculo de dientes
   int                  teeth_shift,      // desplazamiento horizontal de dientes
   int                  lips_period,      // período para el cálculo de labios
   int                  lips_shift,       // desplazamiento horizontal de labios
   ENUM_MA_METHOD       ma_method,        // tipo de suavizado
   ENUM_APPLIED_PRICE   applied_price     // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

jaw_period

[in] Período promedio para la línea azul (mandíbulas de aligátor).

jaw_shift

[in] Desplazamiento de la línea azul con relación al gráfico de precios.

teeth_period

[in] Período promedio para la línea roja (dientes de aligátor).

teeth_shift

[in] Desplazamiento de la línea roja con relación al gráfico de precios.

lips_period

[in] Período promedio para la línea verde (labios de aligátor).

lips_shift

[in] Desplazamiento de la línea verde con relación al gráfico de precios.

ma_method

[in] Método de promedio. Puede ser cualquier valor de la enumeración ENUM_MA_METHOD.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

© 2000-2025, MetaQuotes Ltd.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Nota

Números de búfers: 0 - GATORJAW_LINE, 1 - GATORTEETH_LINE, 2 - GATORLIPS_LINE.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_iAlligator.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iAlligator."
#property description "El símbolo y el período de tiempo en el que se calcula el indic"
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el"
#property description "Todos los demás parámetros son iguales a los del Alligator está

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- plot Jaws
#property indicator_label1  "Jaws"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot Teeth
#property indicator_label2  "Teeth"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot Lips
#property indicator_label3  "Lips"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
enum Creation
  {
   Call_iAlligator,         // usar iAlligator
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iAlligator;   // tipo de función
input string                symbol=" ";             // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // período de tiempo
input int                   jaw_period=13;          // período para la línea de Mandíbu
input int                   jaw_shift=8;            // desplazamiento de la línea de Ma
input int                   teeth_period=8;         // período para la línea de Dientes
input int                   teeth_shift=5;          // desplazamiento de la línea de Di
input int                   lips_period=5;          // período para la línea de Labios
input int                   lips_shift=3;           // desplazamiento de la línea de La
input ENUM_MA_METHOD        MA_method=MODE_SMMA;    // método de promediación de las li
input ENUM_APPLIED_PRICE    applied_price=PRICE_MEDIAN;// tipo del precio partiendo del
//--- búfers indicadores
double          JawsBuffer[];
double          TeethBuffer[];
double          LipsBuffer[];
//--- variable para guardar el manejador del indicador iAlligator
int     handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Alligator
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de arrays a los búfers indicadores
   SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
//--- estableceremos el desplazamiento para cada línea
   PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iAlligator)
      handle=iAlligator(name,period,jaw_period,jaw_shift,teeth_period,
                        teeth_shift,lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[8];
     //--- períodos y desplazamientos de las líneas del Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
//--- tipo de suavizado
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
//--- tipo del precio
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
//--- crearemos el manejador
      handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iAlligator para el par %s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador Al
   short_name=StringFormat("iAlligator(%s/%s, %d,%d,%d,%d,%d,%d)",name,EnumToString(pe
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iAlligator
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array JawsBuffer supera el número de valores en el indicador iAlliga
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador Alligator
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,l
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Alligator
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iAlligator    |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &jaws_buffer[],  // búfer indicador para la línea de
                           int j_shift,            // desplazamiento de la línea de Ma
                           double &teeth_buffer[], // búfer indicador para la línea de
                           int t_shift,            // desplazamiento de la línea de Di
                           double &lips_buffer[],  // búfer indicador para la línea de
                           int l_shift,            // desplazamiento de la línea de La
                           int ind_handle,         // manejador del indicador iAlligat
                           int amount              // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array JawsBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iAlligator, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array TeethBuffer con los valores desde el búfer indicado
   if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iAlligator, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array LipsBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iAlligator, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAMA

Devuelve el manejador del indicador Adaptive Moving Average. Tiene sólo un búfer.

```
int  iAMA(
   string               symbol,             // nombre del símbolo
   ENUM_TIMEFRAMES      period,             // período
   int                  ama_period,         // período AMA
   int                  fast_ma_period,     // período de la media móvil rápida
   int                  slow_ma_period,     // período de la media móvil lenta
   int                  ama_shift,          // desplazamiento horizontal del indicador
   ENUM_APPLIED_PRICE   applied_price       // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ama_period

[in] Período de cálculo durante el cual se calcula el coeficiente de efectividad.

fast_ma_period

[in] Período rápido para el cálculo del coeficiente de suavizado para un mercado rápido.

slow_ma_period

[in] Período lento para el cálculo del coeficiente de suavizado en ausencia de la tendencia.

ama_shift

[in] Desplazamiento del indicador con relación al precio del gráfico.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iAMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iAMA."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los de la AMA estándar

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iAMA
#property indicator_label1  "iAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAMA,              // usar iAMA
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iAMA;          // tipo de función
input string                symbol=" ";              // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;   // período
input int                   ama_period=15;           // período para el cálculo
input int                   fast_ma_period=2;        // período de la MM rápida
input int                   slow_ma_period=30;       // período de la MM lenta
input int                   ama_shift=0;             // desplazamiento horizontal
input ENUM_APPLIED_PRICE    applied_price;           // tipo de precio
//--- búfer indicador
double          iAMABuffer[];
//--- variable para guardar el manejador del indicador iAMA
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Adaptive Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
//--- fijaremos el desplazamiento
   PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iAMA)
      handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,appli
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[5];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ama_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=fast_ma_period;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slow_ma_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ama_shift;
      //--- tipo de precio
      pars[4].type=TYPE_INT;
      pars[4].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_AMA,5,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iAMA para el par %s/%s, c
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador Ada
```

© 2000-2025, MetaQuotes Ltd.

```
   short_name=StringFormat("iAMA(%s/%s,%d,%d,%d,d)",name,EnumToString(period),ama_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iAlligator
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iADBuffer supera el número de valores en el indicador iAD para
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador Alligator
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

© 2000-2025, MetaQuotes Ltd.

```
                              short_name,
                              values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Accelerator Oscillator
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iAMA              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // búfer indicador de la línea AMA
                         int a_shift,           // desplazamiento de la línea AMA
                         int ind_handle,        // manejador del indicador iAMA
                         int amount             // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iAMABuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iAlligator, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAO

Devuelve el manejador del indicador Awesome Oscillator. Tiene sólo un búfer.

```
int  iAO(
   string           symbol,     // nombre del símbolo
   ENUM_TIMEFRAMES  period      // período
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAO.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iAO."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- Construcción de iAO
#property indicator_label1  "iAO"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen,clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAO,               // usar iAO
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iAO;          // tipo de función
input string                symbol=" ";             // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // período de tiempo
//--- búfers indicadores
double          iAOBuffer[];
double          iAOColors[];
//--- variable para guardar el manejador del indicador iAO
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Awesome Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iAOColors,INDICATOR_COLOR_INDEX);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iAO)
      handle=iAO(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AO);
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iAO para el par %s/%s, có
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos para qué par símbolo/período ha sido calculado el indicador Awesome
   short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iAO
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iAOBuffer supera el número de valores en el indicador iAO para
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desc
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays iAOBuffer y iAOColors con los valores desde el indicador Awe
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArraysFromBuffer(iAOBuffer,iAOColors,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Accelerator Oscillator
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iAO           |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // búfer indicador de los valores d
                          double &color_indexes[], // búfer de color (para almacenar l
                          int ind_handle,          // manejador del indicador iAO
                          int amount               // número de valores a copiar
                          )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iAOBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iAO, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- ahora copiaremos los índices de colores
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los valores de los colores desde el indicador iAO,
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iATR

Devuelve el manejador del indicador Average True Range. Tiene sólo un búfer.

```
int  iATR(
   string           symbol,        // nombre del símbolo
   ENUM_TIMEFRAMES  period,        // período
   int              ma_period      // período promedio
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio para el cálculo del indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iATR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iATR."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iATR
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iATR,// usar iATR
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input int                  atr_period=14;          // período para el cálculo
input Creation             type=Call_iATR;          // tipo de función
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // período de tiempo
//--- búfer indicador
double         iATRBuffer[];
//--- variable para guardar el manejador del indicador iAC
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Average True Range
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iATRBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iATR)
      handle=iATR(name,period,atr_period);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=atr_period;
      handle=IndicatorCreate(name,period,IND_ATR,1,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iATR para el par %s/%s, c
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador Ave
   short_name=StringFormat("iATR(%s/%s, period=%d)",name,EnumToString(period),atr_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iATR
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- si el array iATRBuffer supera el número de valores en el indicador iATR pa
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iATRBuffer con los valores desde el indicador Average True Ran
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Accelerator Oscillator
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iATR              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // búfer indicador de valores ATR
                         int ind_handle,    // manejador del indicador iATR
                         int amount         // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iATRBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iATR, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBearsPower

Devuelve el manejador del indicador Bears Power. Tiene sólo un búfer.

```
int  iBearsPower(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  ma_period,         // período promedio
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio para el cálculo del indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBearsPower.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iBearsPower
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Construcción de iBearsPower
#property indicator_label1  "iBearsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBearsPower,       // usar iBearsPower
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iBearsPower;  // tipo de función
input int                  ma_period=13;           // período de la media móvil
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // período de tiempo
//--- búfer indicador
double         iBearsPowerBuffer[];
//--- variable para guardar el manejador del indicador iBearsPower
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Bears Power
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iBearsPower)
      handle=iBearsPower(name,period,ma_period);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- período de la media móvil
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iBearsPower para el par %
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador Bea
   short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),m
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iBearsPower
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- si el array iATRBuffer supera el número de valores en el indicador iBearsP
       //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- significa que no es la primera vez que se calcula nuestro indicador y desd
       //--- se ha añadido no más de una barra para la calculación
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- llenamos el array iBearsPowerBuffer con los valores desde el indicador Bears Pow
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
    if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
    string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
    Comment(comm);
//--- recordaremos el número de valores en el indicador Bears Power
    bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
    return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iBearsPower       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // el búfer indicador de valores Bears Pow
                         int ind_handle,    // manejador del indicador iBearsPower
                         int amount         // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
    ResetLastError();
//--- llenamos una parte del array iBearsPowerBuffer con los valores desde el búfer in
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
      {
       //--- si el proceso de copiado ha fallado, comunicaremos el código del error
       PrintFormat("Fallo al copiar los datos desde el indicador iBearsPower, código de
       //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
       return(false);
      }
//--- todo ha salido bien
    return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBands

Devuelve el manejador del indicador Bollinger Bands®.

```
int  iBands(
   string               symbol,           // nombre del símbolo
   ENUM_TIMEFRAMES      period,           // período
   int                  bands_period,     // período para el cálculo de la línea media
   int                  bands_shift,      // desplazamiento horizontal del indicador
   double               deviation,        // número de desviaciones estándares
   ENUM_APPLIED_PRICE   applied_price     // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

bands_period

[in] Período promedio para la línea principal del indicador.

bands_shift

[in] Desplazamiento del indicador con relación al gráfico de precios.

deviation

[in] Desviación de la línea principal.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Nota

Números de búfers: 0 - BASE_LINE, 1 - UPPER_BAND, 2 - LOWER_BAND

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iBands.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iBands."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- construcción de Upper
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- construcción de Lower
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrMediumSeaGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- construcción de Middle
#property indicator_label3  "Middle"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrMediumSeaGreen
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBands,             // usar iBands
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iBands;          // tipo de función
input int                  bands_period=20;           // período de la media móvil
input int                  bands_shift=0;             // sangría
input double               deviation=2.0;             // número de desviaciones estánd
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de precio
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // período de tiempo
//--- búfers indicadores
double         UpperBuffer[];
double         LowerBuffer[];
```

© 2000-2025, MetaQuotes Ltd.

```
double         MiddleBuffer[];
//--- variable para guardar el manejador del indicador iBands
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Bollinger Bands
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- estableceremos el desplazamiento para cada línea
   PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iBands)
      handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[4];
      //--- período de la media móvil
      pars[0].type=TYPE_INT;
      pars[0].integer_value=bands_period;
      //--- desplazamiento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=bands_shift;
      //--- número de desviaciones estándares
      pars[2].type=TYPE_DOUBLE;
      pars[2].double_value=deviation;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- tipo de precio
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iBands para el par %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos para qué par símbolo/período ha sido calculado el indicador Bollinge
   short_name=StringFormat("iBands(%s/%s, %d,%d,%G,%s)",name,EnumToString(period),
                           bands_period,bands_shift,deviation,EnumToString(applied_pri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iBands
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      //--- si el tamaño de los arrays indicadores supera el número de valores en el i
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array con los valores desde el indicador Bollinger Bands
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle,v
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Bollinger Bands
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iBands        |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &base_values[],     // búfer indicador de la línea r
                           double &upper_values[],    // búfer indicador del borde sup
                           double &lower_values[],    // búfer indicador del borde inf
                           int shift,                 // desplazamiento
                           int ind_handle,            // manejador del indicador iBand
                           int amount                 // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array MiddleBuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iBands, código del err
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- llenamos una parte del array UpperBuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iBands, código del er
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array LowerBuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iBands, código del er
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBullsPower

Devuelve el manejador del indicador Bulls Power. Tiene sólo un búfer.

```
int  iBullsPower(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  ma_period,         // período promedio
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio para el cálculo del indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_iBullsPower.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iBullsPower
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iBullsPower
#property indicator_label1  "iBullsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBullsPower,       // usar iBullsPower
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iBullsPower;  // tipo de función
input int                  ma_period=13;           // período de la media móvil
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // período de tiempo
//--- búfer indicador
double         iBullsPowerBuffer[];
//--- variable para guardar el manejador del indicador iBullsPower
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Bulls Power
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iBullsPowerBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iBullsPower)
      handle=iBullsPower(name,period,ma_period);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- período de la media móvil
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iBullsPower para el par %
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador Bul
   short_name=StringFormat("iBullsPower(%s/%s, period=%d)",name,EnumToString(period),m
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iBullsPower
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      //--- si el array iBullsPowerBuffer supera el número de valores en el indicador
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desc
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iBullsPowerBuffer con los valores desde el indicador Bulls Pow
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iBullsPowerBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Bulls Power
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iBullsPower       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // el búfer indicador de valores Bulls Pow
                         int ind_handle,    // manejador del indicador iBullsPower
                         int amount         // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iBullsPowerBuffer con los valores desde el búfer in
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iBullsPower, código de
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

iCCI

Devuelve el manejador del indicador Commodity Channel Index. Tiene sólo un búfer.

```
int  iCCI(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  ma_period,         // período promedio
   ENUM_APPLIED_PRICE   applied_price      // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio para el cálculo del indicador.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iCCI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iCCI."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- construcción de iCCI
#property indicator_label1  "iCCI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- niveles horizontales en la ventana del indicador
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iCCI,              // usar iCCI
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iCCI;               // tipo de función
input int                   ma_period=14;                 // período de la media móvil
input ENUM_APPLIED_PRICE    applied_price=PRICE_TYPICAL;  // tipo de precio
input string                symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- búfer indicador
double         iCCIBuffer[];
//--- variable para guardar el manejador del indicador iCCI
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Commodity Channel Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iCCI)
      handle=iCCI(name,period,ma_period,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[2];
      //--- período de la media móvil
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo de precio
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_CCI,2,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iCCI para el par %s/%s, c
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos para qué par símbolo/período ha sido calculado el indicador Bollinge
   short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- número de valores copiados desde el indicador iCCI
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iATRBuffer supera el número de valores en el indicador iCCI pa
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iCCIBuffer con los valores desde el indicador Commodity Channe
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Commodity Channel Index
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iCCI              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // búfer indicador de valores Commodity Ch
                         int ind_handle,    // manejador del indicador iCCI
                         int amount         // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- llenamos una parte del array iCCIBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iCCI, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iChaikin

Devuelve el manejador del indicador Chaikin Oscillator. Tiene sólo un búfer.

```
int  iChaikin(
   string               symbol,              // nombre del símbolo
   ENUM_TIMEFRAMES      period,              // período
   int                  fast_ma_period,      // período rápido
   int                  slow_ma_period,      // período lento
   ENUM_MA_METHOD       ma_method,           // tipo de suavizado
   ENUM_APPLIED_VOLUME  applied_volume       // valor utilizado
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

fast_ma_period

[in] Período promedio rápido para el cálculo del indicador.

slow_ma_period

[in] Período promedio lento para el cálculo del indicador.

ma_method

[in] Tipo de promedio. Puede ser una de las constantes del promedio ENUM_MA_METHOD.

applied_volume

[in] Valor utilizado. Puede ser una de las constantes de ENUM_APPLIED_VOLUME.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_iChaikin.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "de los búfers indicadores para el indicador técnico iChaikin."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iChaikin
#property indicator_label1  "iChaikin"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iChaikin,          // usar iChaikin
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iChaikin;           // tipo de función
input int                   fast_ma_period=3;             // período rápido
input int                   slow_ma_period=10;            // período lento
input ENUM_MA_METHOD        ma_method=MODE_EMA;           // tipo de suavizado
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // tipo de volumen
input string                symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- búfer indicador
double         iChaikinBuffer[];
//--- variable para guardar el manejador del indicador iChaikin
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Chaikin Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iChaikin)
      handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_volu
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[4];
      //--- período rápido
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ma_period;
      //--- período lento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ma_period;
      //--- tipo de suavizado
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo de volumen
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iChaikin para el par %s/%
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos para qué par símbolo/período ha sido calculado el indicador Chaikin
   short_name=StringFormat("iChaikin(%s/%s, %d, %d, %s, %s)",name,EnumToString(period)
                           fast_ma_period,slow_ma_period,
                           EnumToString(ma_method),EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iChaikin
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iChaikinBuffer supera el número de valores en el indicador iCh
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iChaikinBuffer con los valores desde el indicador Chaikin Osci
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Chaikin Oscillator
```

© 2000-2025, MetaQuotes Ltd.

```
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iChaikin          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // búfer indicador de valores Chaikin Osci
                         int ind_handle,    // manejador del indicador iChaikin
                         int amount         // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iChaikinBuffer con los valores desde el búfer indic
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iChaikin, código del e
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iCustom

Devuelve el manejador del indicador personalizado especificado.

```
int  iCustom(
   string           symbol,     // nombre del símbolo
   ENUM_TIMEFRAMES  period,     // período
   string           name        // carpeta/nombre_de_indicador personalizado
   ...                          // lista de parámetros de entrada del indicador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

name

[in] Nombre del indicador personalizado. Si antes del nombre indicamos la barra oblicua inversa '\', el archivo EX5 del indicador se buscará con respecto a la carpeta raíz MQL5. De esta forma, al llamar iCustom(Symbol(), Period(), "\FirstIndicator"...), el indicador se cargará como MQL5\FirstIndicator.ex5. Si el archivo no se encuentra según esta ruta, aparecerá el error 4802 (ERR_INDICATOR_CANNOT_CREATE).

Si la ruta no comienza con '\', la búsqueda y la carga del indicador se realizarán en la siguiente secuencia:

- En primer lugar, el archivo EX5 del indicador se buscará en la misma carpeta donde se encuentra el archivo EX5 del programa que realiza la llamada. Por ejemplo, el asesor CrossMA.EX5 se encuentra en la carpeta MQL5\Experts\MyExperts y contiene la llamada iCustom(Symbol(), Period(), "SecondIndicator"...), entonces, la búsqueda del indicador se realizará en la ruta MQL5\Experts\MyExperts\SecondIndicator.ex5.
- Si el indicador no ha sido localizado en el mismo directorio, la búsqueda se realizará con respecto al directorio raíz de indicadores MQL5\Indicators. Es decir, se buscará el archivo MQL5\Indicators\SecondIndicator.ex5. Si el indicador no ha sido localizado en ninguna de las rutas, la función retornará INVALID_HANDLE y se mostrará el error 4802 (ERR_INDICATOR_CANNOT_CREATE).

Si la ruta al indicador se ha establecido en un subdirectorio, por ejemplo, como MyIndicators\ThirdIndicator, la búsqueda primero se efectuará en la carpeta del programa que ha realizado la llamada (el asesor se encuentra en la carpeta MQL5\Experts\MyExperts), en la ruta MQL5\Experts\MyExperts\MyIndicators\ThirdIndicator.ex5, y después, si no ha habido éxito, se buscará el archivo MQL5\Indicators\MyIndicators\ThirdIndicator.ex5. En este caso, además, deberemos indicar como separador en la ruta la barra oblicua inversa doble '\\', por ejemplo, iCustom(Symbol(), Period(), "MyIndicators\\ThirdIndicator"...)

...

© 2000-2025, MetaQuotes Ltd.

[in] input-parámetros del indicador personalizado están separados por comas. El tipo y el orden de seguimiento de parámetros deben corresponder. Si los parámetros no están especificados, entonces se usarán los valores por defecto.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE.

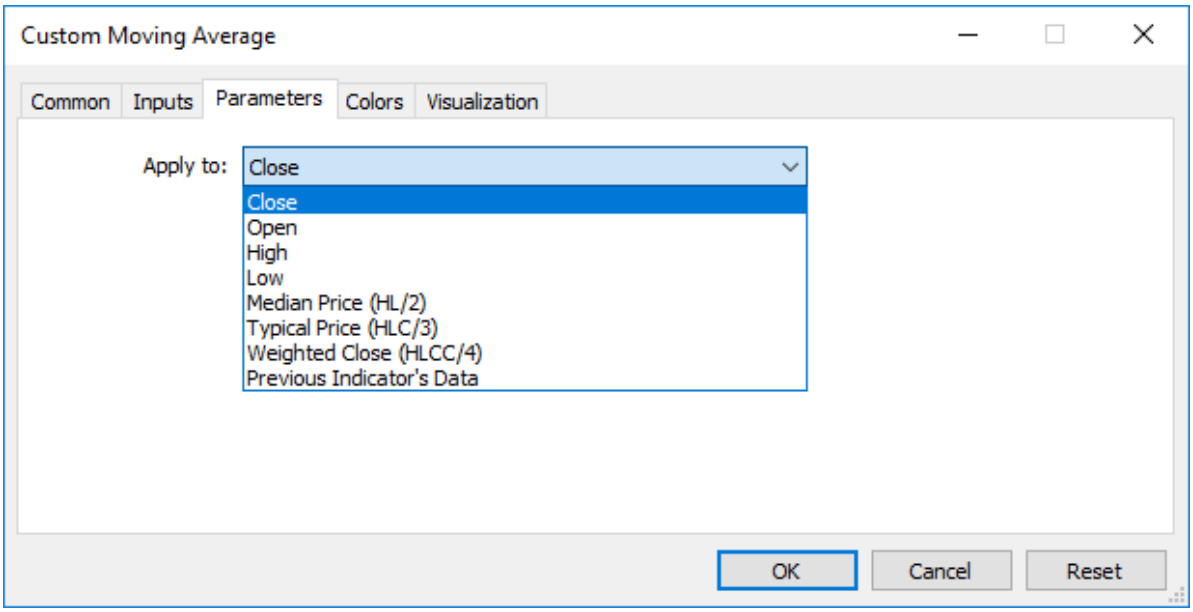
Nota

Un indicador personalizado tiene que estar compilado (un archivo con la extensión EX5) y debe estar ubicado en el directorio MQL5/Indicators del terminal de cliente o en una de sus subcarpetas.

Los indicadores que requieren verificaciones se definen automáticamente desde la llamada a la función iCustom(), si el parámetro correspondiente es fijado por una cadena constante. Para los demás casos (uso de función IndicatorCreate() o uso de una cadena no constante en el parámetro que asigna el nombre del indicador) esta propiedad #property tester_indicator es necesaria:

```
#property tester_indicator "indicator_name.ex5"
```

Si en el indicador se usa la forma de llamada, entonces al iniciar el indicador personalizado en la pestaña "Parameters" podemos especificar adicionalmente los datos a base de los cuales va a ser calculado. Si el parámetro "Apply to" no está elegido explícitamente, por defecto el cálculo se realiza a base de los valores "Close".



Cuando un indicador personalizado se llama desde el programa mql5, el parámetro Applied_Price o un manejador de otro indicador debe ser pasado el último después de todos los parámetros de entrada previstos por el indicador personalizado.

Véase también

Propiedades de programas, Acceso a las series temporales y a los datos de indicadores, IndicatorCreate(), IndicatorRelease()

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int MA_Period=21;
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- indicator buffers
double         Label1Buffer[];
//--- manejador del indicador personalizado Custom Moving Average.mq5
int MA_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
   ResetLastError();
   MA_handle=iCustom(NULL,0,"Examples\\Custom Moving Average",
                     MA_Period,
                     MA_Shift,
                     MA_Method,
                     PRICE_CLOSE // calculamos según los precios de cierre
                     );
   Print("MA_handle = ",MA_handle,"  error = ",GetLastError());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const int &spread[])
  {
//--- vamos a copiar los valores del indicador Custom Moving Average a nuestro búfer c
   int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
   Print("copy = ",copy,"    rates_total = ",rates_total);
//--- si el intento es fallido, avisemos de ello
   if(copy<=0)
      Print("Fallo al obtener los valores del indicador Custom Moving Average");
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

iDEMA

Devuelve el manejador del indicador Double Exponential Moving Average. Tiene sólo un búfer.

```
int  iDEMA(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  ma_period,         // período promedio
   int                  ma_shift,          // desplazamiento horizontal del indicador
   ENUM_APPLIED_PRICE   applied_price      // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio (cantidad de barras) para el cálculo del indicador.

ma_shift

[in] Desplazamiento del indicador con relación al gráfico de precios.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iDEMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iDEMA."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iDEMA
#property indicator_label1  "iDEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDEMA,              // usar iDEMA
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iDEMA;           // tipo de función
input int                   ma_period=14;              // período de la media móvil
input int                   ma_shift=0;                // desplazamiento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo de precio
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double         iDEMABuffer[];
//--- variable para guardar el manejador del indicador iDEMA
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Double Exponential Movi
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
   //--- fijaremos el desplazamiento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iDEMA)
      handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[3];
      //--- período de la media móvil
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- desplazamiento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de precio
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iDEMA para el par %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador Dou
   short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iDEMA
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iDEMABuffer supera el número de valores en el indicador iDEMA
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iDEMABuffer con los valores desde el indicador Double Exponent
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Double Exponential Moving Aver
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iDEMA             |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // búfer indicador de valores Double Expor
                         int shift,         // desplazamiento
                         int ind_handle,    // manejador del indicador iDEMA
                         int amount         // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iDEMABuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iDEMA, código del erro
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iDeMarker

Devuelve el manejador del indicador DeMarker. Tiene sólo un búfer.

```
int  iDeMarker(
   string           symbol,        // nombre del símbolo
   ENUM_TIMEFRAMES  period,        // período
   int              ma_period      // período promedio
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio para el cálculo del indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iDeMarker.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iDeMarker."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iDeMarker
#property indicator_label1  "iDeMarker"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- niveles horizontales en la ventana del indicador
#property indicator_level1  0.3
#property indicator_level2  0.7
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDeMarker,         // usar iDeMarker
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iDeMarker;       // tipo de función
input int                   ma_period=14;              // período de la media móvil
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double          iDeMarkerBuffer[];
//--- variable para guardar el manejador del indicador iDeMarker
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador DeMarker
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iDeMarker)
      handle=iDeMarker(name,period,ma_period);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[1];
      //--- período de la media móvil
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iDeMarker para el par %s/
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador DeM
   short_name=StringFormat("iDeMarker(%s/%s, period=%d)",name,EnumToString(period),ma_
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iDeMarker
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iDEMABuffer supera el número de valores en el indicador iDeMar
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iDeMarkerBuffer con los valores desde el indicador DeMarker
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador DeMarker
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iDeMarker         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // búfer indicador de valores DeMarker
                         int ind_handle,    // manejador del indicador iDeMarker
                         int amount         // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iDeMarkerBuffer con los valores desde el búfer indi
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iDeMarker, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iEnvelopes

Devuelve el manejador del indicador Envelopes.

```
int  iEnvelopes(
   string               symbol,           // nombre del símbolo
   ENUM_TIMEFRAMES      period,           // período
   int                  ma_period,        // período para calcular la línea media
   int                  ma_shift,         // desplazamiento horizontal del indicador
   ENUM_MA_METHOD       ma_method,        // tipo de suavizado
   ENUM_APPLIED_PRICE   applied_price,    // tipo de precio o manejador
   double               deviation         // desviación de márgenes respecto a la líne
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio de la línea principal del indicador.

ma_shift

[in] Desplazamiento del indicador con relación al gráfico de precios.

ma_method

[in] Método de promedio. Puede ser cualquier valor de la enumeración ENUM_MA_METHOD.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

deviation

[in] Desviación de la línea principal en términos de porcientos.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Nota

Números de búfers: 0 - UPPER_LINE, 1 - LOWER_LINE.

Ejemplo:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                  Demo_iEnvelopes.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iEnvelopes."
#property description "El símbolo y el período de tiempo en el que se calcula el indic"
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- construcción de Upper
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- construcción de Lower
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iEnvelopes,        // usar iEnvelopes
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iEnvelopes;      // tipo de función
input int                  ma_period=14;              // período de la media móvil
input int                  ma_shift=0;                // desplazamiento
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // tipo de suavizado
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de precio
input double               deviation=0.1;             // desviación de los márgenes de
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double         UpperBuffer[];
double         LowerBuffer[];
//--- variable para guardar el manejador del indicador iEnvelopes
```

© 2000-2025, MetaQuotes Ltd.

```
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Envelopes
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- estableceremos el desplazamiento para cada línea
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iEnvelopes)
      handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviati
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[5];
      //--- período de la media móvil
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- desplazamiento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de suavizado
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo de precio
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      //--- tipo de precio
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[4].type=TYPE_DOUBLE;
      pars[4].double_value=deviation;
      handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iEnvelopes para el par %s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador Env
   short_name=StringFormat("iEnvelopes(%s/%s, %d, %d, %s,%s, %G)",name,EnumToString(pe
   ma_period,ma_shift,EnumToString(ma_method),EnumToString(applied_price),deviation);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iEnvelopes
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- si el array UpperBuffer supera el número de valores en el indicador iEnvel
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays UpperBuffer y LowerBuffer con los valores desde el indicador
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy))
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Envelopes
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iEnvelopes    |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &upper_values[],    // búfer indicador del borde sup
                           double &lower_values[],    // búfer indicador del borde inf
                           int shift,                 // desplazamiento
                           int ind_handle,            // manejador del indicador iEnve
                           int amount                 // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array UpperBuffer con los valores desde el búfer indicado
   if(CopyBuffer(ind_handle,0,-shift,amount,upper_values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iEnvelopes, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- llenamos una parte del array LowerBuffer con los valores desde el búfer indicado
   if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iEnvelopes, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iForce

Devuelve el manejador del indicador Force Index. Tiene sólo un búfer.

```
int  iForce(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  ma_period,         // período promedio
   ENUM_MA_METHOD       ma_method,         // tipo de suavizado
   ENUM_APPLIED_VOLUME applied_volume      // tipo de volumen para el cálculo
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio para el cálculo del indicador.

ma_method

[in] Método de promedio. Puede ser cualquier valor de la enumeración ENUM_MA_METHOD.

applied_volume

[in] Volumen usado. Puede ser cualquier valor de la enumeración ENUM_APPLIED_VOLUME.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iForce.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iForce."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iForce
#property indicator_label1  "iForce"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iForce,            // usar iForce
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iForce;             // tipo de función
input int                   ma_period=13;                 // período promedio
input ENUM_MA_METHOD        ma_method=MODE_SMA;           // tipo de suavizado
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // tipo de volumen
input string                symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- búfer indicador
double          iForceBuffer[];
//--- variable para guardar el manejador del indicador iForce
int     handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Force
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
       name=_Symbol;
      }
//--- crearemos el manejador del indicador
   if(type==Call_iForce)
      handle=iForce(name,period,ma_period,ma_method,applied_volume);
   else
      {
       //--- llenaremos la estructura con los valores de los parámetros del indicador
       MqlParam pars[3];
       //--- período de la media móvil
       pars[0].type=TYPE_INT;
       pars[0].integer_value=ma_period;
       //--- tipo de suavizado
       pars[1].type=TYPE_INT;
       pars[1].integer_value=ma_method;
       //--- tipo de volumen
       pars[2].type=TYPE_INT;
       pars[2].integer_value=applied_volume;
       //--- tipo de precio
       handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
      }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
      {
       //--- avisaremos sobre el fallo y mostraremos el número del error
       PrintFormat("Fallo al crear el manejador del indicador iForce para el par %s/%s,
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- el trabajo del indicador se finaliza anticipadamente
       return(INIT_FAILED);
      }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador For
   short_name=StringFormat("iForce(%s/%s, %d, %s, %s)",name,EnumToString(period),
                           ma_period,EnumToString(ma_method),EnumToString(applied_volu
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iForce
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iForceBuffer supera el número de valores en el indicador iForc
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desc
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iForceBuffer con los valores desde el indicador Force
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Force
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iForce            |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // búfer indicador de valores Force Index
```

© 2000-2025, MetaQuotes Ltd.

```
                          int ind_handle,     // manejador del indicador iForce
                          int amount          // número de valores a copiar
                          )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iForceBuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iForce, código del err
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iFractals

Devuelve el manejador del indicador Fractals.

```
int  iFractals(
   string           symbol,     // nombre del símbolo
   ENUM_TIMEFRAMES  period      // período
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Nota

Números de búfers: 0 - UPPER_LINE, 1 - LOWER_LINE.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_iFractals.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iFractals."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- construcción de FractalUp
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_label1  "FractalUp"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
//--- construcción de FractalDown
#property indicator_label2  "FractalDown"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFractals,         // usar iFractals
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iFractals;          // tipo de función
input string                symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- búfers indicadores
double          FractalUpBuffer[];
double          FractalDownBuffer[];
//--- variable para guardar el manejador del indicador iFractals
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Fractals
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
//--- estableceremos los códigos desde el conjunto Wingdings para las propiedades PLOT
   PlotIndexSetInteger(0,PLOT_ARROW,217); // flecha arriba
   PlotIndexSetInteger(1,PLOT_ARROW,218); // flecha abajo
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iFractals)
      handle=iFractals(name,period);
   else
      handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iFractals para el par %s/
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador Fra
   short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iFractals
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
```

© 2000-2025, MetaQuotes Ltd.

```
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array FractalUpBuffer supera el número de valores en el indicador iF
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays FractalUpBuffer y FractalDownBuffer con los valores desde el
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy))
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Fractals
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iFractals     |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &up_arrows[],        // búfer indicador de las flech
                           double &down_arrows[],      // búfer indicador de las flech
                           int ind_handle,             // manejador del indicador iFra
                           int amount                  // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array FractalUpBuffer con los valores desde el búfer indi
   if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iFractals al array Fra
                  GetLastError());
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- llenamos una parte del array FractalDownBuffer con los valores desde el búfer in
   if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iFractals al array Fra
                  GetLastError());
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iFrAMA

Devuelve el manejador del indicador Fractal Adaptive Moving Average. Tiene sólo un búfer.

```
int  iFrAMA(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  ma_period,         // período promedio
   int                  ma_shift,          // desplazamiento horizontal del indicador
   ENUM_APPLIED_PRICE   applied_price      // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio (cantidad de barras) para el cálculo del indicador.

ma_shift

[in] Desplazamiento del indicador con relación al gráfico de precios.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iFrAMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iFrAMA."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iFrAMA
#property indicator_label1  "iFrAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFrAMA,            // usar iFrAMA
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iFrAMA;             // tipo de función
input int                   ma_period=14;                 // período promedio
input int                   ma_shift=0;                   // desplazamiento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;    // tipo de precio
input string                symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- búfer indicador
double         iFrAMABuffer[];
//--- variable para guardar el manejador del indicador iFrAMA
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Fractal Adaptive Moving
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
//--- fijaremos el desplazamiento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iFrAMA)
      handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[3];
      //--- período de la media móvil
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- desplazamiento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de precio
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      //--- tipo de precio
      handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iFrAMA para el par %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos a base de qué par símbolo/período ha sido calculado el indicador iFr
   short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2025, MetaQuotes Ltd.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iFrAMA
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iFrAMABuffer supera el número de valores en el indicador iFrAM
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iFrAMABuffer con los valores desde el indicador Fractal Adapti
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Fractal Adaptive Moving Averag
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Llenamos el búfer indicador desde el indicador iFrAMA            |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // búfer indicador de valores Fractal Adap
                         int shift,         // desplazamiento
                         int ind_handle,    // manejador del indicador iFrAMA
                         int amount         // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iFrAMABuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iFrAMA, código del err
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iGator

Devuelve el manejador del indicador Gator. El oscilador muestra la diferencia entre la línea azul y roja del Aligátor (histograma de arriba) y la diferencia entre la línea roja y verde (histograma de abajo).

```
int  iGator(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  jaw_period,        // período para el cálculo de mandíbulas
   int                  jaw_shift,         // desplazamiento horizontal de mandíbulas
   int                  teeth_period,      // período para el cálculo de dientes
   int                  teeth_shift,       // desplazamiento horizontal de dientes
   int                  lips_period,       // período para el cálculo de labios
   int                  lips_shift,        // desplazamiento horizontal de labios
   ENUM_MA_METHOD       ma_method,         // tipo de suavizado
   ENUM_APPLIED_PRICE   applied_price      // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

jaw_period

[in] Período promedio para la línea azul (mandíbulas de aligátor).

jaw_shift

[in] Desplazamiento de la línea azul del Aligátor con relación al gráfico de precios. No tiene relación directa con el desplazamiento visual del histograma del indicador.

teeth_period

[in] Período promedio para la línea roja (dientes de aligátor).

teeth_shift

[in] Desplazamiento de la línea roja del Aligátor con relación al gráfico de precios. No tiene relación directa con el desplazamiento visual del histograma del indicador.

lips_period

[in] Período promedio para la línea verde (labios de aligátor).

lips_shift

[in] Desplazamiento de la línea verde del Aligátor con relación al gráfico de precios. No tiene relación directa con el desplazamiento visual del histograma del indicador.

ma_method

[in] Método de promedio. Puede ser cualquier valor de la enumeración ENUM_MA_METHOD.

applied_price

© 2000-2025, MetaQuotes Ltd.

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Nota

Números de buffers: 0 - UPPER_HISTOGRAM, 1- buffer de color del histograma de arriba, 2 - LOWER_HISTOGRAM, 3- buffer de color del histograma de abajo.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iGator.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iGator."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Gator Oscillat

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
//--- construcción de GatorUp
#property indicator_label1  "GatorUp"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- construcción de GatorDown
#property indicator_label2  "GatorDown"
#property indicator_type2   DRAW_COLOR_HISTOGRAM
#property indicator_color2  clrGreen, clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   Call_iGator,            // usar iGator
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iGator;       // tipo de función
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // período de tiempo
input int                  jaw_period=13;          // período para la línea de Mandíbu
input int                  jaw_shift=8;            // desplazamiento de la línea de Ma
input int                  teeth_period=8;         // período para la línea de Dientes
input int                  teeth_shift=5;          // desplazamiento de la línea de Di
input int                  lips_period=5;          // período para la línea de Labios
input int                  lips_shift=3;           // desplazamiento de la línea de La
input ENUM_MA_METHOD       MA_method=MODE_SMMA;    // método de promediación de las lí
input ENUM_APPLIED_PRICE   applied_price=PRICE_MEDIAN;// tipo del precio partiendo del
//--- búfers indicadores
double         GatorUpBuffer[];
double         GatorUpColors[];
double         GatorDownBuffer[];
double         GatorDownColors[];
//--- variable para guardar el manejador del indicador iGator
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- valores de desplazamiento para el histograma superior e inferior
int shift;
//--- vamos a guardar el número de los valores en el indicador Gator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
   SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);
/*
  ¡Todos los desplazamientos especificados en los parámetros se refieren al indicador
  ¡Por esta razón dichos desplazamientos no mueven el mismo indicador Gator, sino muev
  a base de cuyos valores se construyen los valores del indicador Gator Oscillator!
*/
//--- calcularemos el desplazamiento para el histograma superior e inferior que supone
   shift=MathMin(jaw_shift,teeth_shift);
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//--- a pesar de que el indicador contiene dos histogramas, se utiliza el mismo despla
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(1,PLOT_SHIFT,shift);

//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iGator)
      handle=iGator(name,period,jaw_period,jaw_shift,teeth_period,teeth_shift,
                    lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[8];
      //--- períodos y desplazamientos de las líneas del Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
      //--- tipo de suavizado
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
      //--- tipo de precio
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
      //--- crearemos el manejador
      handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iGator para el par %s/%s,
                  name,
```

© 2000-2025, MetaQuotes Ltd.

```
                  EnumToString(period),
                  GetLastError());
       //--- el trabajo del indicador se finaliza anticipadamente
       return(INIT_FAILED);
      }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Gator (
   short_name=StringFormat("iGator(%s/%s, %d, %d ,%d, %d, %d, %d)",name,EnumToString(p
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iGator
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array GatorUpBuffer supera el número de valores en el indicador iGat
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//--- llenamos los arrays con los valores desde el indicador Gator Oscillator
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownColo
      shift,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Gator Oscillator
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iGator        |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &ups_buffer[],         // búfer indicador para el hi
                           double &up_color_buffer[],    // búfer indicador para los i
                           double &downs_buffer[],       // búfer indicador para el hi
                           double &downs_color_buffer[], // búfer indicador para los i
                           int u_shift,                  // desplazamiento para el his
                           int ind_handle,               // manejador del indicador iG
                           int amount                    // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array GatorUpBuffer con los valores desde el búfer indica
   if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iGator, código del err
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array GatorUpColors con los valores desde el búfer indica
   if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iGator, código del err
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- llenamos una parte del array GatorDownBuffer con los valores desde el búfer indi
   if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iGator, código del err
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array GatorDownColors con los valores desde el búfer indi
   if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iGator, código del err
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iIchimoku

Devuelve el manejador del indicador Ichimoku Kinko Hyo.

```
int  iIchimoku(
   string           symbol,           // nombre del símbolo
   ENUM_TIMEFRAMES  period,           // período
   int              tenkan_sen,       // período Tenkan-sen
   int              kijun_sen,        // período Kijun-sen
   int              senkou_span_b     // período Senkou Span B
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

tenkan_sen

[in] Período promedio Tenkan Sen.

kijun_sen

[in] Período promedio Kijun Sen.

senkou_span_b

[in] Período promedio Senkou Span B.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Nota

Números de búfers: 0 - TENKANSEN_LINE, 1 - KIJUNSEN_LINE, 2 - SENKOUSPANA_LINE, 3 - SENKOUSPANB_LINE, 4 - CHIKOUSPAN_LINE.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_iIchimoku.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "de los búfers indicadores para el indicador técnico iIchimoku."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Ichimoku Kinko

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   4
//--- construcción de Tenkan_sen
#property indicator_label1  "Tenkan_sen"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- construcción de Kijun_sen
#property indicator_label2  "Kijun_sen"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrBlue
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- construcción de Senkou_Span
#property indicator_label3  "Senkou Span A;Senkou Span B" // en la Data Window se most
#property indicator_type3   DRAW_FILLING
#property indicator_color3  clrSandyBrown, clrThistle
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//--- construcción de Chikou_Span
#property indicator_label4  "Chinkou_Span"
#property indicator_type4   DRAW_LINE
#property indicator_color4  clrLime
#property indicator_style4  STYLE_SOLID
#property indicator_width4  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iIchimoku,         // usar iIchimoku
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iIchimoku;       // tipo de función
input int                  tenkan_sen=9;              // período de Tenkan-sen
input int                  kijun_sen=26;              // período de Kijun-sen
input int                  senkou_span_b=52;          // período de Senkou Span B
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
```

© 2000-2025, MetaQuotes Ltd.

```
double         Tenkan_sen_Buffer[];
double         Kijun_sen_Buffer[];
double         Senkou_Span_A_Buffer[];
double         Senkou_Span_B_Buffer[];
double         Chinkou_Span_Buffer[];
//--- variable para guardar el manejador del indicador iIchimoku
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Ichimoku Kinko Hyo
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,Tenkan_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(1,Kijun_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(2,Senkou_Span_A_Buffer,INDICATOR_DATA);
   SetIndexBuffer(3,Senkou_Span_B_Buffer,INDICATOR_DATA);
   SetIndexBuffer(4,Chinkou_Span_Buffer,INDICATOR_DATA);
//--- estableceremos el desplazamiento para el canal Senkou Span a kijun_sen barras en
   PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//--- no hace falta establecer el desplazamiento para la línea Chikou Span, porque los
//--- se guardan en el indicador iIchimoku con un desplazamiento ya fijado
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iIchimoku)
      handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[3];
      //--- períodos y desplazamientos de las líneas del Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=tenkan_sen;
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[1].type=TYPE_INT;
      pars[1].integer_value=kijun_sen;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=senkou_span_b;
      //--- crearemos el manejador
      handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iIchimoku para el par %s/
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Ichimok
   short_name=StringFormat("iIchimoku(%s/%s, %d, %d ,%d)",name,EnumToString(period),
                           tenkan_sen,kijun_sen,senkou_span_b);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iIchimoku
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
```

© 2000-2025, MetaQuotes Ltd.

```
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array Tenkan_sen_Buffer supera el número de valores en el indicador
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador Ichimoku Kinko Hyo
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer,S
      kijun_sen,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Ichimoku Kinko Hyo
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iIchimoku     |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[],     // búfer indicador de la l
                           double &kijun_sen_buffer[],      // búfer indicador de la l
                           double &senkou_span_A_buffer[],  // búfer indicador de la l
                           double &senkou_span_B_buffer[],  // búfer indicador de la l
                           double &chinkou_span_buffer[],   // búfer indicador Chinkou
                           int senkou_span_shift,           // desplazamiento de la lí
                           int ind_handle,                  // manejador del indicador
                           int amount                       // número de valores a cop
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array Tenkan_sen_Buffer con los valores desde el búfer in
   if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("1.Fallo al copiar los datos desde el indicador iGator, código del e
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array Kijun_sen_Buffer con los valores desde el búfer ind
   if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("2.Fallo al copiar los datos desde el indicador iGator, código del e
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array Chinkou_Span_Buffer con los valores desde el búfer
//--- si senkou_span_shift>0 la línea se desplaza hacia el futuro a senkou_span_shift
   if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("3.Fallo al copiar los datos desde el indicador iGator, código del e
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array Senkou_Span_A_Buffer con los valores desde el búfer
//--- si senkou_span_shift>0 la línea se desplaza hacia el futuro a senkou_span_shift
   if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("4.Fallo al copiar los datos desde el indicador iGator, código del e
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array Senkou_Span_B_Buffer con los valores desde el búfer
//--- cuando copiamos Chinkou Span no hace falta considerar el desplazamiento porque l
//--- se guardan en el indicador iIchimoku con un desplazamiento ya fijado
   if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("5.Fallo al copiar los datos desde el indicador iGator, código del e
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBWMFI

Devuelve el manejador del indicador Market Facilitation Index. Tiene sólo un búfer.

```
int  iBWMFI(
   string               symbol,             // nombre del símbolo
   ENUM_TIMEFRAMES      period,             // período
   ENUM_APPLIED_VOLUME  applied_volume      // tipo de volumen para el cálculo
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

applied_volume

[in] Volumen usado. Puede ser cualquier valor de la enumeración ENUM_APPLIED_VOLUME.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iBWMFI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iBWMFI."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- construcción de iBWMFI
#property indicator_label1  "iBWMFI"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrLime,clrSaddleBrown,clrBlue,clrPink
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBWMFI,            // usar iBWMFI
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iBWMFI;          // tipo de función
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;// tipo de volumen
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double          iBWMFIBuffer[];
double          iBWMFIColors[];
//--- variable para guardar el manejador del indicador iBWMFI
int     handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Market Facilitation Inc
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iBWMFI)
      handle=iBWMFI(name,period,applied_volume);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- llenaremos la estructura con los valores de los parámetros del indicador
       MqlParam pars[1];
       //--- tipo de volumen
       pars[0].type=TYPE_INT;
       pars[0].integer_value=applied_volume;
       handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
      }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
      {
       //--- avisaremos sobre el fallo y mostraremos el número del error
       PrintFormat("Fallo al crear el manejador del indicador iIchimoku para el par %s/
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- el trabajo del indicador se finaliza anticipadamente
       return(INIT_FAILED);
      }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Market
   short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iBWMFI
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
      {
       PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
       return(0);
      }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array Tenkan_sen_Buffer supera el número de valores en el indicador
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador Market Facilitation Index
//--- si FillArraysFromBuffer ha devuelto false, significa que los datos aún no están
   if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy)) return
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Market Facilitation Index
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iBWMFI        |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &values[],    // búfer indicador de valores del hist
                           double &colors[],    // búfer indicador de colores del hist
                           int ind_handle,      // manejador del indicador iBWMFI
                           int amount           // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iBWMFIBuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iBWMFI, código del err
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- llenamos una parte del array iBWMFIColors con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iBWMFI, código del err
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMomentum

Devuelve el manejador del indicador Momentum. Tiene sólo un búfer.

```
int  iMomentum(
   string                symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES       period,            // período
   int                   mom_period,        // período promedio
   ENUM_APPLIED_PRICE    applied_price      // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

mom_period

[in] Período promedio (número de barras) para calcular el cambio del precio.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_iMomentum.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iMomentum."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Momentum están

#property indicator_separate_window
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iMomentum
#property indicator_label1  "iMomentum"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMomentum,          // usar iMomentum
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iMomentum;       // tipo de función
input int                   mom_period=14;             // período de momentum
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo de precio
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double          iMomentumBuffer[];
//--- variable para guardar el manejador del indicador iMomentum
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Momentum
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- crearemos el manejador del indicador
   if(type==Call_iMomentum)
      handle=iMomentum(name,period,mom_period,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[2];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=mom_period;
      //--- tipo de precio
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iMomentum para el par %s/
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Momentu
   short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                           mom_period, EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iMomentum
```

© 2000-2025, MetaQuotes Ltd.

```
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata de la primera inicialización de la calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iMomentumBuffer supera el número de valores en el indicador iM
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iMomentumBuffer con los valores desde el indicador Momentum
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Momentum
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iMomentum         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // búfer indicador de valores Momentum
                         int ind_handle,    // manejador del indicador iMomentum
                         int amount         // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iMomentumBuffer con los valores desde el búfer indi
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iMomentum, código del
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMFI

Cálculo de Money Flow Index.

```
int  iMFI(
   string               symbol,             // nombre del símbolo
   ENUM_TIMEFRAMES      period,             // período
   int                  ma_period,          // período promedio
   ENUM_APPLIED_VOLUME  applied_volume      // tipo de volumen para el cálculo
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio (número de barras) para el cálculo del indicador.

applied_volume

[in] Valor utilizado. Puede ser uno de ENUM_APPLIED_VOLUME.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                    Demo_iMFI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iMFI."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Money Flow Inc

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- construcción de iMFI
#property indicator_label1  "iMFI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- niveles horizontales en la ventana del indicador
#property indicator_level1  20
#property indicator_level2  80
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMFI,              // usar iMFI
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iMFI;               // tipo de función
input int                   ma_period=14;                 // período
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // tipo de volumen
input string                symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- búfer indicador
double         iMFIBuffer[];
//--- variable para guardar el manejador del indicador iMFI
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Money Flow Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iMFI)
      handle=iMFI(name,period,ma_period,applied_volume);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[2];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo de volumen
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_MFI,2,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iMFI para el par %s/%s, c
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Money F
   short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period, EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- número de valores copiados desde el indicador iMFI
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iMFIBuffer supera el número de valores en el indicador iMFI so
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iMFIBuffer con los valores desde el indicador Money Flow Index
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Money Flow Index
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iMFI              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // búfer indicador de valores Money Flow I
                         int ind_handle,    // manejador del indicador iMFI
                         int amount         // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- llenamos una parte del array iMFIBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iMFI, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMA

Devuelve el manejador del indicador de la media móvil. Tiene sólo un búfer.

```
int  iMA(
   string                symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES       period,            // período
   int                   ma_period,         // período promedio
   int                   ma_shift,          // desplazamiento horizontal del indicador
   ENUM_MA_METHOD        ma_method,         // tipo de suavizado
   ENUM_APPLIED_PRICE    applied_price      // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio para calcular la media móvil.

ma_shift

[in] Desplazamiento del indicador con relación al gráfico de precios.

ma_method

[in] Método de promedio. Puede obtener cualquier valor de ENUM_MA_METHOD.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iMA."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los de la Moving Avera

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iMA
#property indicator_label1  "iMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMA,               // usar iMA
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iMA;               // tipo de función
input int                  ma_period=10;                // período de la media
input int                  ma_shift=0;                  // desplazamiento
input ENUM_MA_METHOD       ma_method=MODE_SMA;          // tipo de suavizado
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;   // tipo de precio
input string               symbol=" ";                  // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // timeframe
//--- búfer indicador
double         iMABuffer[];
//--- variable para guardar el manejador del indicador iMA
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- fijaremos el desplazamiento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iMA)
      handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[4];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- desplazamiento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de suavizado
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo de precio
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MA,4,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iMA para el par %s/%s, có
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Moving
   short_name=StringFormat("iMA(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period, ma_shift,EnumToString(ma_method),EnumToString(ap
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iMA
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iMABuffer supera el número de valores en el indicador iMA sobr
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array iMABuffer con los valores desde el indicador Moving Average
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Moving Average
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iMA               |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // búfer indicador de valores Moving Aver
                         int shift,          // desplazamiento
                         int ind_handle,     // manejador del indicador iMA
                         int amount          // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iMABuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iMA, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iOsMA

Devuelve el manejador del indicador Moving Average of Oscillator. El oscilador OsMA muestra la diferencia entre los valores del MACD y su línea de señales. Tiene sólo un búfer.

```
int  iOsMA(
   string               symbol,              // nombre del símbolo
   ENUM_TIMEFRAMES      period,              // período
   int                  fast_ema_period,     // período de la media móvil rápida
   int                  slow_ema_period,     // período de la media móvil lenta
   int                  signal_period,       // período promedio para su deferencia
   ENUM_APPLIED_PRICE   applied_price        // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

fast_ema_period

[in] Período medio para calcular la media móvil rápida.

slow_ema_period

[in] Período medio para calcular la media móvil lenta.

signal_period

[in] Período medio para calcular la línea de señales.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Nota

En algunos sistemas este oscilador también se conoce como el histograma MACD.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOsMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iOsMA."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los de la Moving Avera

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iOsMA
#property indicator_label1  "iOsMA"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOsMA,             // usar iOsMA
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iOsMA;           // tipo de función
input int                  fast_ema_period=12;        // período de la media rápida
input int                  slow_ema_period=26;        // período de la media lenta
input int                  signal_period=9;           // período promedio de la difere
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de precio
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double         iOsMABuffer[];
//--- variable para guardar el manejador del indicador iAMA
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Moving Average of Oscil
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iOsMA)
      handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[4];
      //--- período rápido
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- período lento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- período de promedio de la diferencia entre la media lenta y la rápida
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- tipo de precio
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iOsMA para el par %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Moving
   short_name=StringFormat("iOsMA(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
```

© 2000-2025, MetaQuotes Ltd.

```
                              fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iOsMA
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iOsMABuffer supera el número de valores en el indicador iOsMA
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador iOsMA
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

© 2000-2025, MetaQuotes Ltd.

```
                              short_name,
                              values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Moving Average of Oscillator
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iOsMA             |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // búfer indicador de valores OsMA
                         int ind_handle,        // manejador del indicador iOsMA
                         int amount             // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iOsMABuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iOsMA, código del erro
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMACD

Devuelve el manejador del indicador Moving Averages Convergence/Divergence. En los sistemas, donde OsMA lleva el nombre del Histograma MACD, este indicador se muestra en forma de dos líneas. En el terminal de cliente la convergencia/divergencia de las medias móviles se ve en forma de un histograma.

```
int  iMACD(
   string               symbol,              // nombre del símbolo
   ENUM_TIMEFRAMES      period,              // período
   int                  fast_ema_period,     // período de la media móvil rápida
   int                  slow_ema_period,     // período de la media móvil lenta
   int                  signal_period,       // período promedio para su deferencia
   ENUM_APPLIED_PRICE   applied_price        // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

fast_ema_period

[in] Período medio para calcular la media móvil rápida.

slow_ema_period

[in] Período medio para calcular la media móvil lenta.

signal_period

[in] Período medio para calcular la línea de señales.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Nota

Números de búfers: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iMACD.mq5 |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iMACD."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del MACD estándar.

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- construcción de MACD
#property indicator_label1  "MACD"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- construcción de Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_DOT
#property indicator_width2  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMACD,              // usar iMACD
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iMACD;           // tipo de función
input int                   fast_ema_period=12;        // período de la media rápida
input int                   slow_ema_period=26;        // período de la media lenta
input int                   signal_period=9;           // período promedio de la difere
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo de precio
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // período de tiempo
//--- búfers indicadores
double         MACDBuffer[];
double         SignalBuffer[];
//--- variable para guardar el manejador del indicador iMACD
int    handle;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Moving Averages Converg
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iMACD)
      handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[4];
      //--- período rápido
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- período lento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- período de promedio de la diferencia entre la media lenta y la rápida
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- tipo de precio
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MACD,4,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iMACD para el par %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Moving
   short_name=StringFormat("iMACD(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
                           fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iMACD
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array MACDBuffer supera el número de valores en el indicador iMACD s
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- significa que no es la primera vez que se calcula nuestro indicador y desc
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador iMACD
//--- si FillArraysFromBuffers ha devuelto false, significa que los datos no están lis
   if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy)) return(0)
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Moving Averages Convergence/Di
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iMACD         |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &macd_buffer[],    // búfer indicador de valores de
                           double &signal_buffer[],  // búfer indicador de la línea de
                           int ind_handle,           // manejador del indicador iMACD
                           int amount                // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iMACDBuffer con los valores desde el búfer indicado
   if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iMACD, código del erro
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }

//--- llenamos una parte del array SignalBuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iMACD, código del erro
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iOBV

Devuelve el manejador del indicador On Balance Volume. Tiene sólo un búfer.

```
int  iOBV(
   string                symbol,             // nombre del símbolo
   ENUM_TIMEFRAMES       period,             // período
   ENUM_APPLIED_VOLUME   applied_volume      // tipo de volumen para el cálculo
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

applied_volume

[in] Volumen usado. Puede ser cualquier valor de la enumeración ENUM_APPLIED_VOLUME.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOBV.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iOBV."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---  iOBV
#property indicator_label1  "iOBV"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOBV ,             // usar iOBV
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iOBV;               // tipo de función
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // tipo de volumen
input string                symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- búfers indicadores
double          iOBVBuffer[];
//--- variable para guardar el manejador del indicador iOBV
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador On Balance Volume
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iOBV)
      handle=iOBV(name,period,applied_volume);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- tipo de volumen
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_OBV,1,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iOBV para el par %s/%s, c
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador On Bala
   short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iOBV
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
```

© 2000-2025, MetaQuotes Ltd.

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iOBVBuffer supera el número de valores en el indicador iOBV so
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador iOBV
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador On Balance Volume
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iOBV              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &obv_buffer[],  // búfer indicador de valores OBV
                         int ind_handle,        // manejador del indicador iOBV
                         int amount             // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iOBVBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iOBV, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iSAR

Devuelve el manejador del indicador Parabolic Stop and Reverse system. Tiene sólo un búfer.

```
int  iSAR(
   string           symbol,      // nombre del símbolo
   ENUM_TIMEFRAMES  period,      // período
   double           step,        // paso de incremento de velocidad - aceleración
   double           maximum      // coeficiente máximo de seguir un precio
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

step

[in] Aumento del nivel de stop, normalmente - 0.02.

maximum

[in] Nivel máximo de stop, normalmente - 0.2.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iSAR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iSAR."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Parabolic Stop

#property indicator_chart_window
#property indicator_buffers 1
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- construcción de iSAR
#property indicator_label1  "iSAR"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iSAR,              // usar iSAR
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iSAR;              // tipo de función
input double               step=0.02;                   // paso - factor de aceleraci
input double               maximum=0.2;                 // valor máximo del paso
input string               symbol=" ";                  // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // timeframe
//--- búfers indicadores
double         iSARBuffer[];
//--- variable para guardar el manejador del indicador iSAR
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Parabolic SAR
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
//--- estableceremos el código del símbolo desde el conjunto Wingdings para la propiec
   PlotIndexSetInteger(0,PLOT_ARROW,159);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
```

© 2000-2025, MetaQuotes Ltd.

```
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iSAR)
      handle=iSAR(name,period,step,maximum);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[2];
      //--- valor del paso
      pars[0].type=TYPE_DOUBLE;
      pars[0].double_value=step;
      //--- valor límite del paso que se puede utilizar en el cálculo
      pars[1].type=TYPE_DOUBLE;
      pars[1].double_value=maximum;
      handle=IndicatorCreate(name,period,IND_SAR,2,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iSAR para el par %s/%s, c
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Parabol
   short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
                           step,maximum);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- número de valores copiados desde el indicador iSAR
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iSARBuffer supera el número de valores en el indicador iSAR so
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador iSAR
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Parabolic SAR
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iSAR              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &sar_buffer[],  // búfer indicador de valores Paraboli
                         int ind_handle,        // manejador del indicador iSAR
                         int amount             // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- llenamos una parte del array iSARBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iSAR, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iRSI

Devuelve el manejador del indicador Relative Strength Index. Tiene sólo un búfer.

```
int  iRSI(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  ma_period,         // período promedio
   ENUM_APPLIED_PRICE   applied_price      // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio para calcular el índice.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRSI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iRSI."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Relative Stren

#property indicator_separate_window
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iRSI
#property indicator_label1  "iRSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- límites para visualizar valores en la ventana del indicador
#property indicator_maximum 100
#property indicator_minimum 0
//--- niveles horizontales en la ventana del indicador
#property indicator_level1  70.0
#property indicator_level2  30.0
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRSI,              // usar iRSI
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iRSI;               // tipo de función
input int                  ma_period=14;                 // período promedio
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;    // tipo de precio
input string               symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // timeframe
//--- búfer indicador
double         iRSIBuffer[];
//--- variable para guardar el manejador del indicador iRSI
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Relative Strength Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iRSI)
      handle=iRSI(name,period,ma_period,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[2];
      //--- período de la media móvil
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- valor límite del paso que se puede utilizar en el cálculo
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_RSI,2,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iRSI para el par %s/%s, c
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Relativ
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
                           ma_period,applied_price);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iRSI
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iRSIBuffer supera el número de valores en el indicador iRSI so
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array con los valores desde el indicador iRSI
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Relative Strength Index
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iRSI              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rsi_buffer[],  // búfer indicador de valores Relative
                         int ind_handle,        // manejador del indicador iSAR
```

© 2000-2025, MetaQuotes Ltd.

```
                           int amount              // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iRSIBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iRSI, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iRVI

Devuelve el manejador del indicador Relative Vigor Index.

```
int  iRVI(
   string           symbol,        // nombre del símbolo
   ENUM_TIMEFRAMES  period,        // período
   int              ma_period      // período promedio
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio para calcular el índice.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Nota

Números de búfers: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRVI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iRVI."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Relative Vigor

#property indicator_separate_window
#property indicator_buffers 2
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   2
//--- construcción de RVI
#property indicator_label1  "RVI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- construcción de Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRVI,              // usar iRVI
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iRVI;              // tipo de función
input int                  ma_period=10;                // período para el cálculo
input string               symbol=" ";                  // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // timeframe
//--- búfers indicadores
double         RVIBuffer[];
double         SignalBuffer[];
//--- variable para guardar el manejador del indicador iRVI
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Relative Vigor Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iRVI)
      handle=iRVI(name,period,ma_period);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[1];
      //--- período para el cálculo
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_RVI,1,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iRVI para el par %s/%s, c
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Relativ
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- número de valores copiados desde el indicador iRVI
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array RVIBuffer supera el número de valores en el indicador iRVI sob
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador iRVI
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Relative Vigor Index
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iRVI          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rvi_buffer[],     // búfer indicador de valores Relat
                         double &signal_buffer[],  // búfer indicador de la línea de s
                         int ind_handle,           // manejador del indicador iRVI
                         int amount                // número de valores a copiar
                         )
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iRVIBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iRVI, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- llenamos una parte del array SignalBuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iRVI, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iStdDev

Devuelve el manejador del indicador Standard Deviation. Tiene sólo un búfer.

```
int  iStdDev(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  ma_period,         // período promedio
   int                  ma_shift,          // desplazamiento horizontal del indicador
   ENUM_MA_METHOD       ma_method,         // tipo de suavizado
   ENUM_APPLIED_PRICE   applied_price      // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio para el cálculo del indicador.

ma_shift

[in] Desplazamiento del indicador con relación al gráfico de precios.

ma_method

[in] Método de promedio. Puede ser uno de los valores de ENUM_MA_METHOD.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iStdDev.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iStdDev."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Standard Devia

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iStdDev
#property indicator_label1  "iStdDev"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStdDev,           // usar iStdDev
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iStdDev;         // tipo de función
input int                   ma_period=20;              // período promedio
input int                   ma_shift=0;                // desplazamiento
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // tipo de suavizado
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo de precio
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double         iStdDevBuffer[];
//--- variable para guardar el manejador del indicador iStdDev
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Standard Deviation
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- fijaremos el desplazamiento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iStdDev)
      handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[4];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- desplazamiento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de suavizado
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo de precio
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iStdDev para el par %s/%s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Standa
   short_name=StringFormat("iStdDev(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(ma_method),EnumToString(app
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iStdDev
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iStdDevBuffer supera el número de valores en el indicador iStd
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array con los valores desde el indicador Standard Deviation
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Standard Deviation
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iStdDev           |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &std_buffer[],  // búfer indicador de la línea Standa
                         int std_shift,         // desplazamiento de la línea Standar
                         int ind_handle,        // manejador del indicador iStdDev
                         int amount             // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iStdDevBuffer con los valores desde el búfer indic
   if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iStdDev, código del er
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iStochastic

Devuelve el manejador del indicador Stochastic Oscillator.

```
int  iStochastic(
   string           symbol,          // nombre del símbolo
   ENUM_TIMEFRAMES  period,          // período
   int              Kperiod,         // K-período (número de barras para calcular)
   int              Dperiod,         // D-período (período de suavizado inicial)
   int              slowing,         // suavizado final
   ENUM_MA_METHOD   ma_method,       // tipo de suavizado
   ENUM_STO_PRICE   price_field      // modo de cálculo del estocástico
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

Kperiod

[in] K-período (cantidad de barras) para calcular la línea %K.

Dperiod

[in] Período promedio para calcular la línea %D.

slowing

[in] Valor de ralentización.

ma_method

[in] Método de promedio. Puede ser cualquier valor de la enumeración ENUM_MA_METHOD.

price_field

[in] Parámetro de selección de precios para calcular. Puede ser uno de los valores de ENUM_STO_PRICE.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Nota

Números de búfers: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Ejemplo:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                Demo_iStochastic.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iStochastic
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Stochastic Osc

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- construcción de Stochastic
#property indicator_label1  "Stochastic"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- construcción de Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- estableceremos el límite para los valores del indicador
#property indicator_minimum 0
#property indicator_maximum 100
//--- niveles horizontales en la ventana del indicador
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStochastic,       // usar iStochastic
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iStochastic;     // tipo de función
input int                   Kperiod=5;                 // K-período (número de barras p
input int                   Dperiod=3;                 // D-período (período de suaviza
input int                   slowing=3;                 // período para el suavizado fi
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // tipo de suavizado
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_STO_PRICE       price_field=STO_LOWHIGH;   // modo de cálculo del estocásti
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // período de tiempo
//--- búfers indicadores
double         StochasticBuffer[];
double         SignalBuffer[];
//--- variable para guardar el manejador del indicador iStochastic
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Stochastic Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación de los arrays a los búfers indicadores
   SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iStochastic)
      handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[5];
      //--- período K para el cálculo
      pars[0].type=TYPE_INT;
      pars[0].integer_value=Kperiod;
      //--- período D para suavizado inicial
      pars[1].type=TYPE_INT;
      pars[1].integer_value=Dperiod;
      //--- período K para el suavizado final
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slowing;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- tipo de suavizado
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ma_method;
      //--- modo de cálculo del estocástico
      pars[4].type=TYPE_INT;
      pars[4].integer_value=price_field;
      handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iStochastic para el par %
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Stochas
   short_name=StringFormat("iStochastic(%s/%s, %d, %d, %d, %s, %s)",name,EnumToString
                           Kperiod,Dperiod,slowing,EnumToString(ma_method),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iStochastic
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array StochasticBuffer supera el número de valores en el indicador i
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador iStochastic
//--- si FillArraysFromBuffers ha devuelto false, significa que los datos no están lis
   if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy)) ret
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Stochastic Oscillator
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iStochastic   |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &main_buffer[],    // búfer indicador de valores Sto
                           double &signal_buffer[],  // búfer indicador de la línea de
                           int ind_handle,           // manejador del indicador iStoch
                           int amount                // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array StochasticBuffer con los valores desde el búfer ind
   if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iStochastic, código de
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- llenamos una parte del array SignalBuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iStochastic, código de
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iTEMA

Devuelve el manejador del indicador Triple Exponential Moving Average. Tiene sólo un búfer.

```
int  iTEMA(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  ma_period,         // período promedio
   int                  ma_shift,          // desplazamiento horizontal del indicador
   ENUM_APPLIED_PRICE   applied_price      // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio (cantidad de barras) para el cálculo del indicador.

ma_shift

[in] Desplazamiento del indicador con relación al gráfico de precios.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iTEMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iTEMA."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Triple Exponen

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iTEMA
#property indicator_label1  "iTEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTEMA,              // usar iTEMA
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iTEMA;           // tipo de función
input int                   ma_period=14;              // período promedio
input int                   ma_shift=0;                // desplazamiento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo de precio
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double         iTEMABuffer[];
//--- variable para guardar el manejador del indicador iTEMA
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Triple Exponential Movi
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
//--- fijaremos el desplazamiento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iTEMA)
      handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[3];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- desplazamiento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de precio
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iTEMA para el par %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Triple
   short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2025, MetaQuotes Ltd.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iTEMA
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iTEMABuffer supera el número de valores en el indicador iTEMA
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array con los valores desde el indicador Triple Exponential Moving A
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Triple Exponential Moving Aver
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Llenamos el búfer indicador desde el indicador iTEMA             |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &tema_buffer[], // búfer indicador de valores Triple E
                         int t_shift,           // desplazamiento de la línea
                         int ind_handle,        // manejador del indicador iTEMA
                         int amount             // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iTEMABuffer con los valores desde el búfer indicado
   if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iTEMA, código del erro
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iTriX

Devuelve el manejador del indicador Triple Exponential Moving Averages Oscillator. Tiene sólo un búfer.

```
int  iTriX(
   string               symbol,            // nombre del símbolo
   ENUM_TIMEFRAMES      period,            // período
   int                  ma_period,         // período promedio
   ENUM_APPLIED_PRICE   applied_price      // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

ma_period

[in] Período promedio (cantidad de barras) para el cálculo del indicador.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iTriX.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iTriX."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iTriX
#property indicator_label1  "iTriX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTriX,             // usar iTriX
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iTriX;           // tipo de función
input int                  ma_period=14;              // período
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de precio
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double         iTriXBuffer[];
//--- variable para guardar el manejador del indicador iTriX
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Triple Exponential Movi
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- crearemos el manejador del indicador
   if(type==Call_iTriX)
      handle=iTriX(name,period,ma_period,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[2];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo de precio
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iTriX para el par %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Triple
   short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iTriX
```

© 2000-2025, MetaQuotes Ltd.

```
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iTriXBuffer supera el número de valores en el indicador iTriX
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desc
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array con los valores desde el indicador Triple Exponential Moving A
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Triple Exponential Moving Aver
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iTriX             |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &trix_buffer[], // búfer indicador de valores Triple E
                         int ind_handle,        // manejador del indicador iTriX
                         int amount             // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iTriXBuffer con los valores desde el búfer indicado
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iTriX, código del erro
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iWPR

Devuelve el manejador del indicador Larry Williams' Percent Range. Tiene sólo un búfer.

```
int  iWPR(
   string           symbol,          // nombre del símbolo
   ENUM_TIMEFRAMES  period,          // período
   int              calc_period      // período promedio
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

calc_period

[in] Período promedio (cantidad de barras) para el cálculo del indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iWPR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iWPR."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iWPR
#property indicator_label1  "iWPR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrCyan
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- estableceremos el límite para los valores del indicador
#property indicator_minimum -100
#property indicator_maximum 0
//--- niveles horizontales en la ventana del indicador
#property indicator_level1  -20.0
#property indicator_level2  -80.0
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iWPR,              // usar iWPR
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation             type=Call_iWPR;            // tipo de función
input int                  calc_period=14;            // período
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double         iWPRBuffer[];
//--- variable para guardar el manejador del indicador iWPR
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Larry Williams' Percent
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- crearemos el manejador del indicador
   if(type==Call_iWPR)
      handle=iWPR(name,period,calc_period);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[1];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=calc_period;
      handle=IndicatorCreate(name,period,IND_WPR,1,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iWPR para el par %s/%s, c
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Williar
   short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iWPR
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iWPRBuffer supera el número de valores en el indicador iWPR so
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array con los valores desde el indicador Williams' Percent Range
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Williams' Percent Range
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iWPR              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &wpr_buffer[],  // búfer indicador de valores Williams
                         int ind_handle,        // manejador del indicador iWPR
                         int amount             // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iWPRBuffer con los valores desde el búfer indicador
   if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iWPR, código del error
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iVIDyA

Devuelve el manejador del indicador Variable Index Dynamic Average. Tiene sólo un búfer.

```
int  iVIDyA(
   string               symbol,           // nombre del símbolo
   ENUM_TIMEFRAMES      period,           // período
   int                  cmo_period,       // período Chande Momentum
   int                  ema_period,       // período del factor de suavizado
   int                  ma_shift,         // desplazamiento horizontal del indicador
   ENUM_APPLIED_PRICE   applied_price     // tipo de precio o manejador
   );
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo actual.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

cmo_period

[in] Período (cantidad de barras) para el cálculo de Chande Momentum Oscillator.

ema_period

[in] Período (cantidad de barras) EMA para el cálculo del factor de suavizado.

ma_shift

[in] Desplazamiento del indicador con relación al gráfico de precios.

applied_price

[in] Precio que se usa. Puede ser cualquier precio de las constantes de precios ENUM_APPLIED_PRICE o un manejador de otro indicador.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iVIDyA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iVIDyA."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el
#property description "Todos los demás parámetros son iguales a los del Variable Index

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- construcción de iVIDyA
#property indicator_label1  "iVIDyA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVIDyA,            // usar iVIDyA
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iVIDyA;          // tipo de función
input int                   cmo_period=15;             // período Chande Momentum
input int                   ema_period=12;             // período del factor de suaviza
input int                   ma_shift=0;                // desplazamiento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo de precio
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // período de tiempo
//--- búfer indicador
double         iVIDyABuffer[];
//--- variable para guardar el manejador del indicador iVIDyA
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Variable Index Dynamic
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array al búfer indicador
   SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- fijaremos el desplazamiento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iVIDyA)
      handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
   else
     {
      //--- llenaremos la estructura con los valores de los parámetros del indicador
      MqlParam pars[4];
      //--- período Chande Momentum
      pars[0].type=TYPE_INT;
      pars[0].integer_value=cmo_period;
      //--- período del factor de suavizado
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ema_period;
      //--- desplazamiento
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_shift;
      //--- tipo de precio
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
     }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
     {
      //--- avisaremos sobre el fallo y mostraremos el número del error
      PrintFormat("Fallo al crear el manejador del indicador iVIDyA para el par %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- el trabajo del indicador se finaliza anticipadamente
      return(INIT_FAILED);
     }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Triple
   short_name=StringFormat("iVIDyA(%s/%s, %d, %d, %d, %s)",name,EnumToString(period),
                           cmo_period,ema_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iVIDyA
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
      return(0);
     }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iWPRBuffer supera el número de valores en el indicador iVIDyA
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos el array con los valores desde el indicador Variable Index Dynamic Aver
//--- si FillArrayFromBuffer ha devuelto false, significa que los datos no están listo
   if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Variable Index Dynamic Average
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos el búfer indicador desde el indicador iVIDyA            |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &vidya_buffer[],// búfer indicador de valores Variable
                         int v_shift,           // desplazamiento de la línea
                         int ind_handle,        // manejador del indicador iVIDyA
                         int amount             // número de valores a copiar
                         )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iVIDyABuffer con los valores desde el búfer indicad
   if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iVIDyA, código del err
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iVolumes

Devuelve el manejador del indicador de volúmenes. Tiene sólo un búfer.

```
int  iVolumes(
   string                symbol,              // nombre del símbolo
   ENUM_TIMEFRAMES       period,              // período
   ENUM_APPLIED_VOLUME   applied_volume       // tipo de volumen para el cálculo
   )
```

Parámetros

symbol

[in] Símbolo del instrumento financiero cuyos datos serán usados para calcular el indicador. NULL significa el símbolo corriente.

period

[in] Valor del período puede ser uno de los valores de la enumeración ENUM_TIMEFRAMES, 0 significa el timeframe actual.

applied_volume

[in] Volumen usado. Puede ser cualquier valor de la enumeración ENUM_APPLIED_VOLUME.

Valor devuelto

Devuelve el manejador del indicador técnico especificado, en caso del fallo devuelve INVALID_HANDLE. Para liberar la memoria del ordenador del indicador que ya no se utiliza la función IndicatorRelease() a la que se le pasa el manejador (handle) de este indicador.

Ejemplo:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iVolumes.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "El indicador demuestra cómo hay que obtener los datos"
#property description "de los búfers indicadores para el indicador técnico iVolumes."
#property description "El símbolo y el período de tiempo en el que se calcula el indic
#property description "se establecen mediante los parámetros symbol y period."
#property description "El modo de crear el manejador (handle) se establece mediante el

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- construcción de iVolumes
#property indicator_label1  "iVolumes"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  Enumeración de modos de crear el manejador                      |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVolumes,          // usar iVolumes
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parámetros de entrada
input Creation              type=Call_iVolumes;           // tipo de función
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // tipo de volumen
input string                symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- búfers indicadores
double          iVolumesBuffer[];
double          iVolumesColors[];
//--- variable para guardar el manejador del indicador iVolumes
int    handle;
//--- variable para guardar
string name=symbol;
//--- nombre del indicador en el gráfico
string short_name;
//--- vamos a guardar el número de los valores en el indicador Volumes
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vinculación del array a los búfers indicadores
   SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
//--- determinamos el símbolo para el que se construye el indicador
   name=symbol;
//--- eliminaremos los espacios del lado izquierdo y derecho
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si después de eso la longitud de la cadena name obtiene el valor cero,
   if(StringLen(name)==0)
     {
      //--- cogeremos el símbolo del gráfico en el que está iniciado el indicador
      name=_Symbol;
     }
//--- crearemos el manejador del indicador
   if(type==Call_iVolumes)
      handle=iVolumes(name,period,applied_volume);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- llenaremos la estructura con los valores de los parámetros del indicador
       MqlParam pars[1];
       //--- tipo de precio
       pars[0].type=TYPE_INT;
       pars[0].integer_value=applied_volume;
       handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
      }
//--- si no se puede crear el manejador
   if(handle==INVALID_HANDLE)
      {
       //--- avisaremos sobre el fallo y mostraremos el número del error
       PrintFormat("Fallo al crear el manejador del indicador iVolumes para el par %s/%
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- el trabajo del indicador se finaliza anticipadamente
       return(INIT_FAILED);
      }
//--- mostraremos sobre qué par símbolo/período ha sido calculado el indicador Volumes
   short_name=StringFormat("iVolumes(%s/%s, %s)",name,EnumToString(period),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialización correcta del indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados desde el indicador iVolumes
   int values_to_copy;
//--- vamos a averiguar el número de valores calculados en el indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
      {
       PrintFormat("BarsCalculated() ha devuelto %d, el código del error %d",calculated
       return(0);
      }
//--- si se trata del primer arranque del proceso de calculación de nuestro indicador
```

© 2000-2025, MetaQuotes Ltd.

```
//--- o si es necesario calcular el indicador para dos o más barras (entonces algo se
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si el array iVolumesBuffer supera el número de valores en el indicador iVo
      //--- en caso contrario, copiaremos menos que el tamaño de los búfers indicadore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- significa que no es la primera vez que se calcula nuestro indicador y desd
      //--- se ha añadido no más de una barra para la calculación
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- llenamos los arrays con los valores desde el indicador iVolumes
//--- si FillArraysFromBuffers ha devuelto false, significa que los datos no están lis
   if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy)) ret
//--- creamos el mensaje
   string comm=StringFormat("%s ==>  Actualizados los valores en el indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- mostraremos en el gráfico un mensaje de servicio
   Comment(comm);
//--- recordaremos el número de valores en el indicador Volumes
   bars_calculated=calculated;
//--- devolveremos el valor prev_calculated para la siguiente llamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Llenamos los búfers indicadores desde el indicador iVolumes      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &volume_buffer[],   // búfer indicador de valores Vo
                           double &color_buffer[],    // búfer indicador de colores
                           int ind_handle,            // manejador del indicador iVolu
                           int amount                 // número de valores a copiar
                           )
  {
//--- actualizaremos el código del error
   ResetLastError();
//--- llenamos una parte del array iVolumesBuffer con los valores desde el búfer indic
   if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iVolumes, código del e
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- llenamos una parte del array iVolumesColors con los valores desde el búfer indic
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
     {
      //--- si el proceso de copiado ha fallado, comunicaremos el código del error
      PrintFormat("Fallo al copiar los datos desde el indicador iVolumes, código del e
      //--- finalizaremos con el resultado nulo - eso quiere decir que el indicador se
      return(false);
     }
//--- todo ha salido bien
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpiaremos el gráfico tras eliminar el indicador
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

Trabajo con resultados de optimización

Aquí tenemos las funciones que sirven para organizar el procesamiento personalizado de los resultados de optimización en el Probador de estrategias. Se puede llamarlas durante la optimización en los agentes de pruebas, así como de forma local en los EAs y scripts.

Cuando Usted arranca un EA en el Probador de estrategias, puede crear su propio array de datos a base de los tipos simples o estructuras simples (no contienen cadenas, objetos de la clase o objetos del array dinámico). Utilizando la función FrameAdd(), Usted puede guardar este conjunto de datos en una estructura especial que se llama frame (cuadro). Durante la optimización de un EA, cada agente puede enviar al terminal una serie de frames. Todos los frames recibidos, en el orden que vayan llegando de los agentes, se escriben en el archivo *.MQD en la carpeta directorio_del_terminal/MQL5/Files/Tester con el nombre del EA. La llegada de los frames al Terminal de Cliente de parte de un agente de pruebas genera el evento TesterPass.

Los frames se puede almacenar tanto en la memoria del ordenador, como en un archivo con el nombre especificado. Por parte del lenguaje MQL5 no existe limitación alguna respecto al número de los frames.

Limitaciones de memoria y espacio en el disco en MQL5 Cloud Network

Al ejecutar la optimización en MQL5 Cloud Network, existe un límite: el asesor experto probado no puede escribir más de 4GB de datos en el disco y utilizar más de 4GB de RAM. Si se excede el límite, el agente de red no podrá completar los cálculos correctamente y usted no obtendrá el resultado de la prueba. En este caso, además, se le cobrará el tiempo ya empleado en los cálculos.

Si necesita obtener información de cada pasada de optimización, utilice el envío de frames sin escribir en el disco. En otras palabras, al realizar cálculos en MQL5 Cloud Network no utilice operaciones de archivo en los asesores durante la optimización, puede utilizar esta comprobación:.

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

Función	Acción
FrameFirst	Mueve el puntero de lectura de frames al inicio y reinicia el filtro establecido antes
FrameFilter	Establece el filtro de lectura de frames y mueve el puntero al inicio
FrameNext	Lee el frame y mueve el puntero al siguiente

© 2000-2025, MetaQuotes Ltd.

Función	Acción
FrameInputs	Recibe los parámetros input sobre los que está formado el frame
FrameAdd	Añade un frame con datos
ParameterGetRange	Recibe para la variable input la información sobre la banda de valores y el paso de cambios durante la optimización del EA en el Probador de Estrategias
ParameterSetRange	Establece las reglas del uso de la variable input durante la optimización del EA en el Probador de Estrategias: valor, paso de cambio, valor inicial y final

Véase también

Estadística de simulación, Información sobre el programa MQL5 en ejecución

© 2000-2025, MetaQuotes Ltd.

FrameFirst

Mueve el puntero de lectura de frames al inicio y reinicia el filtro establecido.

```
bool  FrameFirst();
```

Valor devuelto

Devuelve true en caso del éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

FrameFilter

Establece el filtro de lectura de frames y mueve el puntero al inicio

```
bool  FrameFilter(
   const string  name,          // nombre público/etiqueta
   long          id             // id público
   );
```

Valor devuelto

Devuelve true en caso del éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Si una cadena vacía se pasa como el primer parámetro, el filtro va a trabajar sólo con el parámetro numérico. Es decir se van a ver todos los frames con id especificado. Si el valor del segundo parámetro es igual a ULONG_MAX, trabaja sólo el filtro de texto.

La llamada a FrameFilter("", ULONG_MAX) equivale a la llamada a FrameFirst(), es decir, equivale a la ausencia del filtro.

© 2000-2025, MetaQuotes Ltd.

FrameNext

Lee el frame actual y mueve el puntero al siguiente. Hay 2 variantes de esta función.

1. Llamar para recibir un valor numérico

```
bool  FrameNext(
   ulong&   pass,      // número del paso en la optimización durante el cual ha sido a
   string&  name,      // nombre público/etiqueta
   long&    id,        // id público
   double&  value      // valor
   );
```

2. Llamar para recibir todos los datos del frame

```
bool  FrameNext(
   ulong&   pass,      // número del paso en la optimización durante el cual ha sido a
   string&  name,      // nombre público/etiqueta
   long&    id,        // id público
   double&  value,     // valor
   void&    data[]     // array de cualquier tipo
   );
```

Parámetros

pass

[out] Número del paso durante la optimización en el Probador de estrategias.

name

[out] Nombre del identificador.

id

[out] Valor del identificador.

value

[out] Valor numérico individual.

data

[out] Array de cualquier tipo.

Valor devuelto

Devuelve true en caso del éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Si usa la segunda opción de la llamada, necesita procesar de forma correcta los datos recibidos en el array *data[]*.

© 2000-2025, MetaQuotes Ltd.

FrameInputs

Recibe los parámetros input sobre los que está formado el frame con el número de paso especificado.

```
bool  FrameInputs(
   ulong    pass,                // número del paso en la optimización
   string&  parameters[],        // array de cadenas del tipo "parameterN=valueN"
   uint&    parameters_count     // número total de parámetros
   );
```

Parámetros

pass

[in] Número del paso durante la optimización en el Probador de estrategias.

parameters

[out] Array de cadenas con la descripción de los nombres y valores de los parámetros

parameters_count

[out] Número de elementos que contiene el array *parameters[]*.

Valor devuelto

Devuelve true en caso del éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Una vez recibido el número de cadenas *parameters_count* en el array *parameters[]*, se puede organizar el ciclo para el repaso de todas las entradas. Esto permite conocer los valores de los parámetros de entrada del EA para el número establecido del paso.

© 2000-2025, MetaQuotes Ltd.

FrameAdd

Añade un frame con datos. Hay 2 variantes de esta función.

1. Añadir datos desde un archivo

```
bool  FrameAdd(
   const string  name,        // nombre público/etiqueta
   long          id,          // id público
   double        value,       // valor
   const string  filename     // nombre del archivo con datos
   );
```

2. Añadir datos desde un array de cualquier tipo

```
bool  FrameAdd(
   const string  name,        // nombre público/etiqueta
   long          id,          // id público
   double        value,       // valor
   const void&   data[]       // array de cualquier tipo
   );
```

Parámetros

name

[in] Etiqueta pública frame. Se puede utilizarla para el filtro en la función FrameFilter().

id

[in] Identificador público del frame. Se puede utilizarlo para el filtro en la función FrameFilter().

value

[in] Valor numérico para escribir en el frame. Sirve para transmitir un solitario resultado del paso como en la función OnTester().

filename

[in] Nombre del archivo que contiene los datos para agregar al frame. El archivo debe ubicarse en la carpeta MQL5/Files.

data

[in] Array de cualquier tipo para la escritura en el frame. Se pasa por referencia.

Valor devuelto

Devuelve true en caso del éxito, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

ParameterGetRange

Recibe para la variable input la información sobre la banda de valores y el paso de cambios durante la optimización del EA en el Probador de Estrategias. Hay 2 variantes de esta función.

1. Obtención de la información para el parámetro input del tipo entero

```
bool  ParameterGetRange(
   const string  name,           // nombre del parámetro (variable input)
   bool&         enable,         // permitida la optimización del parámetro
   long&         value,          // valor del parámetro
   long&         start,          // valor inicial
   long&         step,           // paso de cambio
   long&         stop            // valor final
   );
```

2. Obtención de la información para el parámetro input del tipo real

```
bool  ParameterGetRange(
   const string  name,           // nombre del parámetro (variable input)
   bool&         enable,         // permitida la optimización del parámetro
   double&       value,          // valor del parámetro
   double&       start,          // valor inicial
   double&       step,           // paso de cambio
   double&       stop            // valor final
   );
```

Parámetros

name

[in] Identificador de la variable input. Estas variables son parámetros externos del programa cuyos valores pueden ser establecidos durante el arranque en el gráfico o bien durante la simulación.

enable

[out] Quiere decir que este parámetro se puede utilizar para el repaso de valores en el proceso de optimización en el Probador de Estrategias.

value

[out] Valor del parámetro.

start

[out] Valor inicial del parámetro durante la optimización.

step

[out] Paso de cambio del parámetro durante el repaso de sus valores.

stop

[out] Valor final del parámetro durante la optimización.

Valor devuelto

© 2000-2025, MetaQuotes Ltd.

Devuelve true en caso de la ejecución con éxito, de lo contrario - false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Esta función puede ser invocada sólo desde los manejadores OnTesterInit(), OnTesterPass() y OnTesterDeinit(). Sirve para obtener el valor y el rango de cambio de los parámetros de entrada del EA durante el proceso de optimización en el Probador de Estrategias.

Cuando se llama en OnTesterInit(), la información obtenida se puede utilizar para redefinir las reglas de repaso de cualquier variable input a través de la función ParameterSetRange(). De esta manera, se puede establecer nuevos valores Start, Stop, Step, e incluso excluir completamente este parámetro de la optimización a pesar de los ajustes en el probador. Esto permite crear sus propios guiones de manejo del área de parámetros de entrada durante la optimización. Es decir, excluir de la optimización unos parámetros en función de los valores de los parámetros claves del EA.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#property description "Asesor Experto para demostrar la función ParameterGetRange()."
#property description "Hay que iniciar en el Probador de Estrategias en el modo de opt
//--- input parameters
input int                  Input1=1;
input double               Input2=2.0;
input bool                 Input3=false;
input ENUM_DAY_OF_WEEK     Input4=SUNDAY;

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- EA sirve sólo para trabajar en el probador
   if(!MQL5InfoInteger(MQL5_OPTIMIZATION))
     {
      MessageBox("¡Hay que iniciar en el Probador de Estrategias en el modo de optimiz
      //--- finalizamos anticipadamente el trabajo del EA y lo quitamos del gráfico
      return(INIT_FAILED);
     }
//--- inicialización finalizada con éxito
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| TesterInit function                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- ejemplo para el parámetro input del tipo long
   string name="Input1";
   bool enable;
   long par1,par1_start,par1_step,par1_stop;
   ParameterGetRange(name,enable,par1,par1_start,par1_step,par1_stop);
   Print("El primer parámetro");
   PrintFormat("%s=%d  enable=%s  from %d to %d with step=%d",
               name,par1,(string)enable,par1_start,par1_stop,par1_step);
//--- ejemplo para el parámetro input del tipo double
   name="Input2";
   double par2,par2_start,par2_step,par2_stop;
   ParameterGetRange(name,enable,par2,par2_start,par2_step,par2_stop);
   Print("El segundo parámetro");
   PrintFormat("%s=%G  enable=%s  from %G to %G with step=%G",
               name,par2,(string)enable,par2_start,par2_stop,par2_step);

//--- ejemplo para el parámetro input del tipo bool
   name="Input3";
   long par3,par3_start,par3_step,par3_stop;
   ParameterGetRange(name,enable,par3,par3_start,par3_step,par3_stop);
   Print("El tercer parámetro");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,(string)par3,(string)enable,
               (string)par3_start,(string)par3_stop);
//--- ejemplo para el parámetro input del tipo enumeración
   name="Input4";
   long par4,par4_start,par4_step,par4_stop;
   ParameterGetRange(name,enable,par4,par4_start,par4_step,par4_stop);
   Print("El cuarto parámetro");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,EnumToString((ENUM_DAY_OF_WEEK)par4),(string)enable,
               EnumToString((ENUM_DAY_OF_WEEK)par4_start),
               EnumToString((ENUM_DAY_OF_WEEK)par4_stop));
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| TesterDeinit function                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- este mensaje se mostrará una vez finalizada la optimización
   Print(__FUNCTION__," Optimisation completed");
  }
```

© 2000-2025, MetaQuotes Ltd.

ParameterSetRange

Establece las reglas del uso de la variable input durante la optimización del EA en el Probador de Estrategias: valor, paso de cambio, valor inicial y final. Hay 2 variantes de esta función.

1. Establecer los valores para el parámetro input del tipo entero

```
bool  ParameterSetRange(
   const string  name,           // nombre del parámetro (variable input)
   bool          enable,         // permitir la optimización del parámetro
   long          value,          // valor del parámetro
   long          start,          // valor inicial
   long          step,           // paso de cambio
   long          stop            // valor final
   );
```

2. Establecer los valores para el parámetro input del tipo real

```
bool  ParameterSetRange(
   const string  name,           // nombre del parámetro (variable input)
   bool          enable,         // permitir la optimización del parámetro
   double        value,          // valor del parámetro
   double        start,          // valor inicial
   double        step,           // paso de cambio
   double        stop            // valor final
   );
```

Parámetros

name

[in] Identificador de la variable input o sinput. Estas variables son parámetros externos del programa cuyos valores pueden ser establecidos durante el arranque.

enable

[in] Permitir este parámetro para el repaso de valores durante el proceso de optimización en el Probador de Estrategias.

value

[in] Valor del parámetro.

start

[in] Valor inicial del parámetro durante la optimización.

step

[in] Paso de cambio del parámetro durante el repaso de sus valores.

stop

[in] Valor final del parámetro durante la optimización.

Valor devuelto

© 2000-2025, MetaQuotes Ltd.

Devuelve true en caso de la ejecución con éxito, de lo contrario - false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Esta función puede ser invocada sólo desde el manejador OnTesterInit() durante el inicio de optimización en el Probador de Estrategias. Sirve para establecer el rango y paso de cambio del parámetro, pudiendo también excluir completamente este parámetro del proceso de optimización a pesar de los ajustes en el probador. Además, permite utilizar en la optimización incluso las variables declaradas con el modificador sinput.

La función ParameterSetRange() permite crear sus propios guiones de optimización del EA en el probador en función de los valores de los parámetros claves. Es decir, incluir o excluir de la optimización los parámetros de entrada necesarios, así como establecer el rango y paso de cambio necesarios.

© 2000-2025, MetaQuotes Ltd.

Trabajo con eventos

Este grupo contiene las funciones para trabajar con los eventos personalizados y con los eventos de temporizador. Además de estas funciones, existen también funciones especiales para manejar los eventos predefinidos.

Función	Acción
EventSetMillisecondTimer	Arranca el generador de eventos del temporizador de alta precisión con el período inferior a 1 segundo para el gráfico actual
EventSetTimer	Arranca el generador de eventos de temporizador con la periodicidad especificada para el gráfico actual
EventKillTimer	Detiene la generación de eventos en el gráfico actual según el temporizador
EventChartCustom	Genera los eventos personalizados para el gráfico especificado

Véase también

Tipos de eventos del gráfico

© 2000-2025, MetaQuotes Ltd.

EventSetMillisecondTimer

Esta función indica al Terminal de Cliente que los eventos del temporizador para este EA o indicador deben generarse con una periodicidad inferior a un segundo.

```
bool  EventSetMillisecondTimer(
   int  milliseconds      // cantidad de milisegundos
   );
```

Parámetros

milliseconds

[in] Cantidad de milisegundos que determina la frecuencia de los eventos del temporizador.

Valor devuelto

En caso de la ejecución exitosa devuelve true, de lo contrario false. Para obtener el código del error, hay que llamar a la función GetLastError().

Nota

Esta función sirve para los casos cuando se requiere un temporizador de alta resolución. Es decir, cuando es necesario obtener los eventos del temporizador con más frecuencia que una vez al segundo. Si un temporizador con el período más de 1 segundo es suficiente para Ustes, utilice la función EventSetTimer().

En general, con la disminución del período del temporizador se aumenta el tiempo de simulación, puesto que sube el número de llamadas al manejador de eventos del temporizador. Cuando se trabaja en tiempo real, los eventos del temporizador se generan con una frecuencia no más de 1 vez en cada 10-16 milisegundos, esto está relacionado con las limitaciones de hardware.

Por lo común, esta función debe llamarse desde la función OnInit() o en el constructor de la clase. Un EA o un indicador tiene que tener la función OnTimer() para poder manejar los eventos que llegan desde el temporizador.

Cada EA y cada indicador trabaja con su propio temporizador y recibe los eventos sólo de él. Cuando un programa mql5 finaliza su trabajo, el temporizador se destruye de manera forzada en caso si ha sido creado pero no ha sido desactivado por la función EventKillTimer().

Se puede iniciar sólo un temporizador para cada programa. Cada programa mql5 y cada gráfico tienen su propia cola de eventos en la que se colocan todos los eventos según vayan llegando. Si en la cola ya hay un evento Timer, o este evento se encuentra en el proceso de procesamiento, entonces el nuevo evento Timer no se coloca en la cola del programa mql5.

© 2000-2025, MetaQuotes Ltd.

EventSetTimer

Esta función indica al terminal de cliente que hay que generar los eventos desde el temporizador con la periodicidad especificada para este Asesor Experto o el indicador.

```
bool  EventSetTimer(
   int  seconds      // cantidad de segundos
   );
```

Parámetros

seconds

[in] Cantidad de segundos que determina la frecuencia de aparición de eventos desde el temporizador.

Valor devuelto

En caso de éxito devuelve true, de lo contrario devuelve false. Para obtener el código de error hay que llamar a la función GetLastError().

Nota

Habitualmente esta función debe invocarse desde la función OnInit() o en un constructor de clase. Para manejar los eventos que vienen del temporizador, el Asesor Experto o el indicador debe tener la función OnTimer().

Cada Asesor Experto, igual que cada indicador, trabaja con su propio temporizador y recibe eventos únicamente de él. Una vez finalizado el trabajo de un programa mql5, el temporizador se elimina forzadamente, si ha sido creado pero no ha sido desactivado por la función EventKillTimer().

Para cada programa se puede arrancar no más de un temporizador. Cada programa mql5 y cada gráfico tiene su propia cola de eventos en la que se ponen todos los eventos recién llegados. Si en la cola ya hay un evento Timer, o este evento se encuentra en el proceso de tramitación, entonces el nuevo evento Timer no se coloca en la cola del programa mql5.

© 2000-2025, MetaQuotes Ltd.

EventKillTimer

Esta función indica al terminal de cliente que hace falta detener la generación de eventos desde el temporizador para este Asesor Experto o el indicador.

```
void  EventKillTimer();
```

Valor devuelto

No hay valor devuelto.

Nota

Normalmente esta función debe invocarse desde la función OnDeinit() en el caso si en la función OnInit() ha sido invocada la función EventSetTimer(). O debe invocarse del destructor de clase, si en el constructor de esta clase se llama a la función EventSetTimer().

Cada Asesor Experto y cada indicador trabaja con su propio temporizador y recibe eventos únicamente de él. Una vez finalizado el trabajo de un programa mql5, el temporizador se elimina forzadamente, si ha sido creado pero no ha sido desactivado por la función EventKillTimer().

© 2000-2025, MetaQuotes Ltd.

EventChartCustom

Genera un evento personalizado para el gráfico especificado.

```
bool  EventChartCustom(
   long    chart_id,            // identificador del gráfico que recibe el evento
   ushort  custom_event_id,     // identificador del evento
   long    lparam,              // parámetro del tipo long
   double  dparam,              // parámetro del tipo double
   string  sparam               // parámetro literal del evento
   );
```

Parámetros

chart_id

[in] identificador del gráfico. 0 significa el gráfico actual.

custom_event_id

[in] Identificador del evento personalizado. Este identificador se añade automáticamente al valor CHARTEVENT_CUSTOM y se convierte al tipo entero.

lparam

[in] Parámetro del evento del tipo long que se pasa a la función OnChartEvent.

dparam

[in] Parámetro del evento del tipo double que se pasa a la función OnChartEvent.

sparam

[in] Parámetro del evento del tipo string que se pasa a la función OnChartEvent. Si la cadena tiene más de 63 caracteres, esta cadena se recorta.

Valor devuelto

Devuelve true si un evento personalizado ha sido colocado con éxito a la cola de los eventos del gráfico-receptor del evento. En caso del error, devuelve false. Para obtener el código del error, utilice la función GetLastError().

Nota

El Asesor Experto o indicador que está adjuntado al gráfico procesa este evento utilizando la función OnChartEvent(int event_id, long& lparam, double& dparam, string& sparam).

Para cada tipo del evento, los parámetros de entrada de la función OnChartEvent() tienen determinados valores que son necesarios para procesar este evento. En la tabla de abajo están especificados los eventos y valores que se pasan a través de los parámetros.

Evento	Valor del parámetro id	Valor del parámetro lparam	Valor del parámetro dparam	Valor del parámetro sparam
Evento de un teclazo	CHARTEVENT_KEYDOWN	código de la tecla pulsada	Número de pulsaciones de la tecla generadas	Valor literal de la máscara de bits que describe el

© 2000-2025, MetaQuotes Ltd.

Evento	Valor del parámetro id	Valor del parámetro lparam	Valor del parámetro dparam	Valor del parámetro sparam
			mientras ésta se mantenía en estado pulsado	estatus de las teclas del teclado
Eventos del ratón (si para el gráfico está establecida la propiedad CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	coordinada X	coordinada Y	Valor literal de la máscara de bits que describe el estatus de los botones del ratón
Evento de creación de un objeto gráfico (si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_CREATE=true)	CHARTEVENT_OBJECT_CREATE	—	—	Nombre del objeto gráfico creado
Evento del cambio de propiedades de un objeto a través del diálogo de propiedades	CHARTEVENT_OBJECT_CHANGE	—	—	Nombre del objeto gráfico modificado
Evento de eliminación de un objeto gráfico (si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_DELETE=true)	CHARTEVENT_OBJECT_DELETE	—	—	Nombre del objeto gráfico eliminado
Evento de cliquear sobre un gráfico	CHARTEVENT_CLICK	coordinada X	coordinada Y	—
Evento de cliquear sobre un	CHARTEVENT_OBJECT_CLICK	coordinada X	coordinada Y	Nombre del objeto gráfico en

© 2000-2025, MetaQuotes Ltd.

Evento	Valor del parámetro id	Valor del parámetro lparam	Valor del parámetro dparam	Valor del parámetro sparam
objeto gráfico				el que ha ocurrido el evento
Evento de mover un objeto gráfico con el ratón	CHARTEVENT_OBJECT_DRAG	—	—	Nombre del objeto gráfico desplazado
Evento del fin de edición del texto en el campo de introducción del objeto gráfico "Campo de texto"	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nombre del objeto gráfico "Campo de texto" donde ha finalizado la introducción del texto
Evento de modificación del gráfico	CHARTEVENT_CHART_CHANGE	—	—	—
Identificador del evento de usuario	CHARTEVENT_CUSTOM+N	Valor determinado por la función EventChartCustom()	Valor determinado por la función EventChartCustom()	Valor determinado por la función EventChartCustom()

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                            ButtonClickExpert.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

string buttonID="Button";
string labelID="Info";
int broadcastEventID=5000;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- creamos un botón para pasar los eventos de usuario
   ObjectCreate(0,buttonID,OBJ_BUTTON,0,100,100);
   ObjectSetInteger(0,buttonID,OBJPROP_COLOR,clrWhite);
   ObjectSetInteger(0,buttonID,OBJPROP_BGCOLOR,clrGray);
   ObjectSetInteger(0,buttonID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_XSIZE,200);
   ObjectSetInteger(0,buttonID,OBJPROP_YSIZE,50);
   ObjectSetString(0,buttonID,OBJPROP_FONT,"Arial");
   ObjectSetString(0,buttonID,OBJPROP_TEXT,"Botón");
   ObjectSetInteger(0,buttonID,OBJPROP_FONTSIZE,10);
   ObjectSetInteger(0,buttonID,OBJPROP_SELECTABLE,0);

//--- creamos una etiqueta para visualizar la información
   ObjectCreate(0,labelID,OBJ_LABEL,0,100,100);
   ObjectSetInteger(0,labelID,OBJPROP_COLOR,clrRed);
   ObjectSetInteger(0,labelID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,labelID,OBJPROP_YDISTANCE,50);
   ObjectSetString(0,labelID,OBJPROP_FONT,"Trebuchet MS");
   ObjectSetString(0,labelID,OBJPROP_TEXT,"No hay información");
   ObjectSetInteger(0,labelID,OBJPROP_FONTSIZE,20);
   ObjectSetInteger(0,labelID,OBJPROP_SELECTABLE,0);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ObjectDelete(0,buttonID);
   ObjectDelete(0,labelID);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- comprobamos el evento de apretar el botón del ratón
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      string clickedChartObject=sparam;
      //--- si hacemos clic en el objeto con el nombre buttonID
      if(clickedChartObject==buttonID)
        {
         //--- estado del botón - está apretado o no
         bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
         //--- registramos el mensaje de la depuración
         Print("Botón apretado = ",selected);
         int customEventID; // número del evento de usuario a enviar
         string message;    // mensaje a enviar en el evento
         //--- si el botón está apretado
         if(selected)
           {
            message="Botón apretado";
            customEventID=CHARTEVENT_CUSTOM+1;
           }
         else // botón no está apretado
           {
            message="Botón suelto";
            customEventID=CHARTEVENT_CUSTOM+999;
           }
         //--- enviamos un evento de usuario a "nuestro" gráfico
         EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
         //--- enviamos el mensaje a todos los gráficos abiertos
         BroadcastEvent(ChartID(),0,"Broadcast Message");
         //--- mensaje de reparación de errores
         Print("Evento enviado con ID = ",customEventID);
        }
      ChartRedraw();// volvemos a dibujar por vía forzada todos los objetos del gráfic
     }

//--- comprobamos un evento a su pertenencia a eventos de usuario
   if(id>CHARTEVENT_CUSTOM)
     {
      if(id==broadcastEventID)
        {
         Print("Recibido el mensaje de difusión del gráfico con id = "+lparam);
        }
      else
        {
         //--- vamos a leer un mensaje de texto en el evento
         string info=sparam;
         Print("Se procesa el evento de USUARIO con ID = ",id);
         //--- mostramos el mensaje en una etiqueta
         ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
         ChartRedraw();// volvemos a dibujar por vía forzada todos los objetos del gra
        }
     }
  }
//+------------------------------------------------------------------+
//| enviar un mensaje de difusión a todos los gráficos abiertos      |
//+------------------------------------------------------------------+
void BroadcastEvent(long lparam,double dparam,string sparam)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
   long currChart=ChartFirst();
   int i=0;
   while(i<CHARTS_MAX)           // seguramente tenemos no más de CHARTS_MAX de gráfic
     {
      EventChartCustom(currChart,eventID,lparam,dparam,sparam);
      currChart=ChartNext(currChart); // a base del anterior obtenemos un gráfico nuev
      if(currChart==-1) break;        // llegamos al final de la lista de gráficos
      i++;                            // No olvidemos a aumentar el contador
     }
  }
//+------------------------------------------------------------------+
```

Véase también

Eventos de terminal de cliente, Funciones de procesamiento de eventos

© 2000-2025, MetaQuotes Ltd.

Trabajo con OpenCL

Los programas en OpenCL sirven para ejecutar los cálculos en las tarjetas gráficas con el soporte del estándar OpenCL 1.1 o superior. Las tarjetas gráficas modernas cuentan con centenares de pequeños procesadores especializados que son capaces de realizar simultáneamente las operaciones matemáticas sencillas con los flujos de datos entrantes. El lenguaje OpenCL se encarga de estos cálculos paralelos y permite alcanzar una aceleración increíble para algunos tipos de tareas.

En algunas tarjetas gráficas, por defecto, está desactivado el modo de trabajo con los números del tipo double lo que provoca el error de compilación 5105. Para activar el modo del soporte de los números double, hay que insertar la siguiente directiva en el texto del programa OpenCL: #pragma OPENCL EXTENSION cl_khr_fp64 : enable. No obstante, si la tarjeta gráfica no soporta double, la inserción de esta directiva no servirá de nada.

Es recomendable escribir el código fuente de OpenCL en archivos CL aparte, que después se puedan incluir en el programa MQL5 con la ayuda de las variables de recurso.

Procesamiento de archivos en programas OpenCL

Para obtener la información sobre el último error en un programa OpenCL, se usarán las funciones CLGetInfoInteger y CLGetInfoString, que permiten obtener el código de error y la descripción textual.

Código de error OpenCL: para obtener el último error OpenCL, use la llamada CLGetInfoInteger, en este caso, además, el parámetro *handle* será ignorado, es decir, no tendrá ningún valor (se puede indicar cero). Descripción de errores: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS.

Para un código de error desconocido, se retornará la línea "unknown OpenCL error N", donde N será el código de error. Ejemplo:

```
//--- el primer parámetro de handle se ignorará al obtener el último código de error
int code = (int)CLGetInfoInteger(0,CL_LAST_ERROR);
```

Mensaje textual del error OpenCL: para obtenr el último error OpenCL, use la llamada CLGetInfoString, en este caso, además, como parámetro *handle* se transmitirá el código del error.

Descripción del error: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Si se transmite CL_LAST_ERROR en lugar del código de error, la función retornará la descripción del último error. Ejemplo:

```
//--- obtener el último código de error de OpenCL
int    code = (int)CLGetInfoInteger(0,CL_LAST_ERROR);
string desc; // para obtener una descripción de texto del error

//--- use el código de error para obtener una descripción textual del error
if(!CLGetInfoString(code,CL_ERROR_DESCRIPTION,desc))
   desc = "cannot get OpenCL error description, " + (string)GetLastError();
Print(desc);

//--- para obtener la descripción del último error de OpenCL sin obtener primero el có
if(!CLGetInfoString(CL_LAST_ERROR,CL_ERROR_DESCRIPTION, desc))
```

© 2000-2025, MetaQuotes Ltd.

```
   desc = "cannot get OpenCL error description, " + (string)GetLastError();
Print(desc);;
```

Hasta ahora, como descripción del error se daba el nombre de la enumeración interna, cuya decodificación se puede ver en la página. https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Por ejemplo, si se obtiene el valor CL_INVALID_KERNEL_ARGS, la descripción del mismo será "Returned when enqueuing a kernel when some kernel arguments have not been set or are invalid."

Funciones para ejecutar programas en OpenCL:

Función	Acción
CLHandleType	Devuelve el tipo de manejador OpenCL como un valor de la enumeración ENUM_OPENCL_HANDLE_TYPE
CLGetInfoInteger	Devuelve el valor de una propiedad de números enteros para el objeto o dispositivo OpenCL
CLContextCreate	Crea el contexto OpenCL
CLContextFree	Elimina el contexto OpenCL
CLGetDeviceInfo	Recibe la propiedad del dispositivo desde el controlador de dispositivo OpenCL
CLProgramCreate	Crea un programa OpenCL desde el código fuente
CLProgramFree	Elimina un programa OpenCL
CLKernelCreate	Crea la función del arranque OpenCL
CLKernelFree	Elimina la función del arranque OpenCL
CLSetKernelArg	Establece un parámetro para la función OpenCL
CLSetKernelArgMem	Establece el búfer OpenCL como el parámetro de la función OpenCL
CLSetKernelArgMemLocal	Define el búfer local como argumento de la función núcleo
CLBufferCreate	Crea el búfer OpenCL
CLBufferFree	Elimina el búfer OpenCL
CLBufferWrite	Escribe un array en el búfer OpenCL
CLBufferRead	Lee el búfer OpenCL en un array
CLExecute	Ejecuta un programa OpenCL
CLExecutionStatus	Retorna el estado de ejecución del programa OpenCL

Véase también

OpenCL, Recursos

© 2000-2025, MetaQuotes Ltd.

CLHandleType

Devuelve el tipo de manejador OpenCL como un valor de la enumeración ENUM_OPENCL_HANDLE_TYPE.

```
ENUM_OPENCL_HANDLE_TYPE  CLHandleType(
   int  handle     // manejador del objeto OpenCL
   );
```

Parámetros

handle

[in] Manejador para el objeto OpenCL: contexto, kernel, búfer o un programa OpenCL.

Valor devuelto

Tipo del manejador OpenCL como un valor de la enumeración ENUM_OPENCL_HANDLE_TYPE.

ENUM_OPENCL_HANDLE_TYPE

Identificador	Descripción
OPENCL_INVALID	Manejador incorrecto
OPENCL_CONTEXT	Manejador del contexto OpenCL
OPENCL_PROGRAM	Manejador del programa OpenCL
OPENCL_KERNEL	Manejador del kernel OpenCL
OPENCL_BUFFER	Manejador del búfer OpenCL

© 2000-2025, MetaQuotes Ltd.

CLGetInfoInteger

Devuelve el valor de una propiedad de números enteros para el objeto o dispositivo OpenCL.

```
long  CLGetInfoInteger(
   int  handle,                          // manejador del objeto OpenCL o el número d
   ENUM_OPENCL_PROPERTY_INTEGER  prop    // propiedad solicitada
   );
```

Parámetros

handle

[in] Manejador para el objeto OpenCL o el número del dispositivo OpenCL. La numeración de los dispositivos OpenCL se empieza desde cero

prop

[in] Tipo de la propiedad solicitada desde la enumeración ENUM_OPENCL_PROPERTY_INTEGER cuyo valor hay que recibir.

Valor devuelto

Valor de la propiedad especificada en caso del éxito, o -1 en caso del error. Para obtener la información sobre el error, utilice la función GetLastError().

ENUM_OPENCL_PROPERTY_INTEGER

Identificador	Descripción	Tipo
CL_DEVICE_COUNT	El número de dispositivos con soporte de OpenCL. Para esta propiedad no hace falta indicar el primer parámetro, es decir, se puede pasar el valor cero para el parámetro *handle*.	int
CL_DEVICE_TYPE	Tipo del dispositivo	ENUM_CL_DEVICE_TYPE
CL_DEVICE_VENDOR_ID	Identificador único del fabricante	uint
CL_DEVICE_MAX_COMPUTE_UNITS	Número de tareas paralelas calculadas en el dispositivo OpenCL. Un grupo de trabajo se encarga de una tarea computacional. El valor mínimo es igual a 1	uint
CL_DEVICE_MAX_CLOCK_FREQUENCY	Frecuencia máxima establecida del dispositivo en MHz.	uint
CL_DEVICE_GLOBAL_MEM_SIZE	Tamaño de la memoria global del dispositivo en bytes	ulong

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción	Tipo
CL_DEVICE_LOCAL_MEM_SIZE	Tamaño de la memoria local de datos procesados (escenarios) en bytes	uint
CL_BUFFER_SIZE	Tamaño real del búfer OpenCL en bytes	ulong
CL_DEVICE_MAX_WORK_GROUP_SIZE	Número total de grupos locales de trabajo disponibles para un dispositivo OpenCL	ulong
CL_KERNEL_WORK_GROUP_SIZE	Número total de grupos locales de trabajo disponibles para un programa OpenCL	ulong
CL_KERNEL_LOCAL_MEM_SIZE	Tamaño de la memoria local en bytes usada por un programa OpenCL para todas las tareas simultáneas en el grupo. Use CL_DEVICE_LOCAL_MEM_SIZE para obtener el máximo disponible	ulong
CL_KERNEL_PRIVATE_MEM_SIZE	Tamaño mínimo de la memoria privada en bytes usado por cada tarea en el núcleo del programa OpenCL	ulong
CL_LAST_ERROR	Valor del último error de OpenCL	int

La enumeración ENUM_CL_DEVICE_TYPE contiene los posibles tipos de dispositivos con el soporte de OpenCL. Puede obtener el tipo del dispositivo según su número o el handle del objeto OpenCL llamando a CLGetInfoInteger(handle_or_deviceN, CL_DEVICE_TYPE).

ENUM_CL_DEVICE_TYPE

Identificador	Descripción
CL_DEVICE_ACCELERATOR	Acelerador especializado OpenCL (por ejemplo, IBM CELL Blade).
CL_DEVICE_CPU	Uso de la CPU del ordenador como un dispositivo OpenCL. La CPU puede tener uno o más núcleos de cómputo.
CL_DEVICE_GPU	Un dispositivo OpenCL a base de una tarjeta de vídeo.
CL_DEVICE_DEFAULT	Dispositivo OpenCL por defecto. Un dispositivo CL_DEVICE_TYPE_CUSTOM no puede ser el

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
	dispositivo predefinido.
CL_DEVICE_CUSTOM	Los aceleradores especializados que no soportan los programas en OpenCL C.

Ejemplo:

```
void OnStart()
  {
   int cl_ctx;
//--- inicialización del contexto OpenCL
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL not found");
      return;
     }
//--- Visualizamos la información general sobre el dispositivo OpenCL
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY)," MHz"
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE)," by
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE)," byte
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

© 2000-2025, MetaQuotes Ltd.

CLGetInfoString

Esta función devuelve el valor literal de la propiedad para un objeto o dispositivo OpenCL.

```
bool  CLGetInfoString(
   int  handle,                          // manejador del objeto OpenCL o el número d
   ENUM_OPENCL_PROPERTY_STRING  prop,    // propiedad solicitada
   string&  value                        // cadena por referencia
   );
```

Parámetros

handle

[in] Manejador para un objeto OpenCL o número del dispositivo OpenCL. La numeración de los dispositivos OpenCL se empieza desde cero.

prop

[in] Tipo de la propiedad solicitada desde la enumeración ENUM_OPENCL_PROPERTY_STRING cuyo valor hay que obtener.

value

[out] Cadena para obtener el valor de la propiedad.

Valor devuelto

true para la ejecución con éxito, false en caso del error. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

ENUM_OPENCL_PROPERTY_STRING

Identificador	Descripción
CL_PLATFORM_PROFILE	CL_PLATFORM_PROFILE - Perfil OpenCL. El nombre del perfil puede ser uno de los siguientes valores: • FULL_PROFILE - la implementación soporta OpenCL (la funcionalidad está definida como una parte de la especificación del núcleo y no requiere las extensiones adicionales para el soporte de OpenCL); • EMBEDDED_PROFILE - la implementación soporta OpenCL como un suplemento. El perfil enmendado se define como un subconjunto para cada versión OpenCL.
CL_PLATFORM_VERSION	Versión OpenCL
CL_PLATFORM_VENDOR	Nombre del fabricante del dispositivo
CL_PLATFORM_EXTENSIONS	Lista de extensiones que soporta la plataforma. Todos los dispositivos relacionados con esta plataforma deben soportar los nombres de las extensiones
CL_DEVICE_NAME	Nombre del dispositivo

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
CL_DEVICE_VENDOR	Nombre del fabricante
CL_DRIVER_VERSION	Versión del controlador OpenCL en el formato major_number.minor_number
CL_DEVICE_PROFILE	Perfil del dispositivo OpenCL. El nombre del perfil puede ser uno de los siguientes valores: • FULL_PROFILE - la implementación soporta OpenCL (la funcionalidad está definida como una parte de la especificación del núcleo y no requiere las extensiones adicionales para el soporte de OpenCL); • EMBEDDED_PROFILE - la implementación soporta OpenCL como un suplemento. El perfil enmendado se define como un subconjunto para cada versión OpenCL.
CL_DEVICE_VERSION	Versión OpenCL en el formato "OpenCL<space><major_version.minor_version><space><vendor-specific information>"
CL_DEVICE_EXTENSIONS	Lista de extensiones que soporta el dispositivo. La lista puede incluir extensiones soportadas por el fabricante y contener uno o más nombres aprobados: cl_khr_int64_base_atomics cl_khr_int64_extended_atomics cl_khr_fp16 cl_khr_gl_sharing cl_khr_gl_event cl_khr_d3d10_sharing cl_khr_dx9_media_sharing cl_khr_d3d11_sharing
CL_DEVICE_BUILT_IN_KERNELS	Lista de kernels que soporta el dispositivo separados por ";" .
CL_DEVICE_OPENCL_C_VERSION	La versión máxima que soporta el compilador para este dispositivo. Formato de la versión: "OpenCL<space>C<space><major_version.minor_version><space><vendor-specific information> "
CL_ERROR_DESCRIPTION	Descripción textual del error de OpenCL

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   int cl_ctx;
   string str;
//--- inicialización del contexto OpenCL
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL not found");
      return;
     }
//--- Visualizamos la información sobre la plataforma
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_NAME,str))
      Print("OpenCL platform name: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VENDOR,str))
      Print("OpenCL platform vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VERSION,str))
      Print("OpenCL platform ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_PROFILE,str))
      Print("OpenCL platform profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_EXTENSIONS,str))
      Print("OpenCL platform ext: ",str);
//--- Visualizamos la información sobre el dispositivo
   if(CLGetInfoString(cl_ctx,CL_DEVICE_NAME,str))
      Print("OpenCL device name: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_PROFILE,str))
      Print("OpenCL device profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_BUILT_IN_KERNELS,str))
      Print("OpenCL device kernels: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_EXTENSIONS,str))
      Print("OpenCL device ext: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VENDOR,str))
      Print("OpenCL device vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VERSION,str))
      Print("OpenCL device ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_OPENCL_C_VERSION,str))
      Print("OpenCL open c ver: ",str);
//--- Visualizamos la información general sobre el dispositivo OpenCL
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY));
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE));
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE));
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

© 2000-2025, MetaQuotes Ltd.

CLContextCreate

Crea el contexto OpenCL y devuelve el manejador para él.

```
int  CLContextCreate(
   int  device=CL_USE_ANY     // número ordinal del dispositivo OpenCL o macro
   );
```

Parámetros

device

[in] Número del dispositivo OpenCL en el sistema según el orden. En vez de un número concreto se puede indicar uno de los valores:

- CL_USE_ANY - se permite utilizar cualquier dispositivo disponible con el soporte de OpenCL;
- CL_USE_CPU_ONLY - se permite utilizar sólo la emulación OpenCL en CPU;
- CL_USE_GPU_ONLY - la emulación OpenCL está prohibida y se permite utilizar sólo los dispositivos especializados con el soporte de OpenCL (tarjetas gráficas);
- CL_USE_GPU_DOUBLE_ONLY - solo se permite usar GPUs que soportan el tipo double.

Valor devuelto

Manejador para el contexto OpenCL en caso del éxito, o -1 en caso del error. Para obtener la información sobre el error, utilice la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLContextFree

Elimina el contexto OpenCL.

```
void  CLContextFree(
   int  context     // manejador para el contexto OpenCL
   );
```

Parámetros

context

[in] Manejador del contexto OpenCL.

Valor devuelto

Ninguno. En caso de un error interno, se cambia el valor de _LastError. Para obtener la información sobre el error, llame a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLGetDeviceInfo

Recibe la propiedad del dispositivo desde el controlador de dispositivo OpenCL.

```
bool  CLGetDeviceInfo(
   int     handle,          // manejador del dispositivo OpenCL
   int     property_id,     // identificador de la propiedad solicitada
   uchar&  data[],          // array para recibir datos
   uint&   size             // desplazamiento en el array en elementos, por defecto es
   );
```

Parámetros

handle

[in] Número del dispositivo OpenCL o el manejador de OpenCL creado por la función CLContextCreate().

property_id

[in] Identificador de la propiedad que hay que recibir sobre el dispositivo OpenCL. Puede ser uno de los valores predeterminados que se listan en la tabla de abajo.

data[]

[out] Array para recibir los datos sobre la propiedad solicitada.

size

[out] Tamaño de datos recibidos en el array *data[]*.

Valor devuelto

true para la ejecución con éxito, false en caso del error. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

Para los arrays unidimensionales el número del elemento con el que se empieza la lectura de datos para su escritura en el buffer OpenCL se calcula teniendo en cuenta la bandera AS_SERIES.

Lista de disponibles identificadores de propiedades del dispositivo OpenCL

Puede encontrar la descripción de la propiedad más detallada y su función en la página oficial de OpenCL.

Identificador	Valor
CL_DEVICE_TYPE	0x1000
CL_DEVICE_VENDOR_ID	0x1001
CL_DEVICE_MAX_COMPUTE_UNITS	0x1002
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS	0x1003
CL_DEVICE_MAX_WORK_GROUP_SIZE	0x1004
CL_DEVICE_MAX_WORK_ITEM_SIZES	0x1005

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR	0x1006
CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT	0x1007
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT	0x1008
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG	0x1009
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT	0x100A
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE	0x100B
CL_DEVICE_MAX_CLOCK_FREQUENCY	0x100C
CL_DEVICE_ADDRESS_BITS	0x100D
CL_DEVICE_MAX_READ_IMAGE_ARGS	0x100E
CL_DEVICE_MAX_WRITE_IMAGE_ARGS	0x100F
CL_DEVICE_MAX_MEM_ALLOC_SIZE	0x1010
CL_DEVICE_IMAGE2D_MAX_WIDTH	0x1011
CL_DEVICE_IMAGE2D_MAX_HEIGHT	0x1012
CL_DEVICE_IMAGE3D_MAX_WIDTH	0x1013
CL_DEVICE_IMAGE3D_MAX_HEIGHT	0x1014
CL_DEVICE_IMAGE3D_MAX_DEPTH	0x1015
CL_DEVICE_IMAGE_SUPPORT	0x1016
CL_DEVICE_MAX_PARAMETER_SIZE	0x1017
CL_DEVICE_MAX_SAMPLERS	0x1018
CL_DEVICE_MEM_BASE_ADDR_ALIGN	0x1019
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE	0x101A
CL_DEVICE_SINGLE_FP_CONFIG	0x101B
CL_DEVICE_GLOBAL_MEM_CACHE_TYPE	0x101C
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE	0x101D
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE	0x101E
CL_DEVICE_GLOBAL_MEM_SIZE	0x101F
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE	0x1020

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor
CL_DEVICE_MAX_CONSTANT_ARGS	0x1021
CL_DEVICE_LOCAL_MEM_TYPE	0x1022
CL_DEVICE_LOCAL_MEM_SIZE	0x1023
CL_DEVICE_ERROR_CORRECTION_SUPPORT	0x1024
CL_DEVICE_PROFILING_TIMER_RESOLUTION	0x1025
CL_DEVICE_ENDIAN_LITTLE	0x1026
CL_DEVICE_AVAILABLE	0x1027
CL_DEVICE_COMPILER_AVAILABLE	0x1028
CL_DEVICE_EXECUTION_CAPABILITIES	0x1029
CL_DEVICE_QUEUE_PROPERTIES	0x102A
CL_DEVICE_NAME	0x102B
CL_DEVICE_VENDOR	0x102C
CL_DRIVER_VERSION	0x102D
CL_DEVICE_PROFILE	0x102E
CL_DEVICE_VERSION	0x102F
CL_DEVICE_EXTENSIONS	0x1030
CL_DEVICE_PLATFORM	0x1031
CL_DEVICE_DOUBLE_FP_CONFIG	0x1032
CL_DEVICE_PREFERRED_VECTOR_WIDTH_HALF	0x1034
CL_DEVICE_HOST_UNIFIED_MEMORY	0x1035
CL_DEVICE_NATIVE_VECTOR_WIDTH_CHAR	0x1036
CL_DEVICE_NATIVE_VECTOR_WIDTH_SHORT	0x1037
CL_DEVICE_NATIVE_VECTOR_WIDTH_INT	0x1038
CL_DEVICE_NATIVE_VECTOR_WIDTH_LONG	0x1039
CL_DEVICE_NATIVE_VECTOR_WIDTH_FLOAT	0x103A
CL_DEVICE_NATIVE_VECTOR_WIDTH_DOUBLE	0x103B
CL_DEVICE_NATIVE_VECTOR_WIDTH_HALF	0x103C
CL_DEVICE_OPENCL_C_VERSION	0x103D
CL_DEVICE_LINKER_AVAILABLE	0x103E
CL_DEVICE_BUILT_IN_KERNELS	0x103F

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor
CL_DEVICE_IMAGE_MAX_BUFFER_SIZE	0x1040
CL_DEVICE_IMAGE_MAX_ARRAY_SIZE	0x1041
CL_DEVICE_PARENT_DEVICE	0x1042
CL_DEVICE_PARTITION_MAX_SUB_DEVICES	0x1043
CL_DEVICE_PARTITION_PROPERTIES	0x1044
CL_DEVICE_PARTITION_AFFINITY_DOMAIN	0x1045
CL_DEVICE_PARTITION_TYPE	0x1046
CL_DEVICE_REFERENCE_COUNT	0x1047
CL_DEVICE_PREFERRED_INTEROP_USER_SYNC	0x1048
CL_DEVICE_PRINTF_BUFFER_SIZE	0x1049
CL_DEVICE_IMAGE_PITCH_ALIGNMENT	0x104A
CL_DEVICE_IMAGE_BASE_ADDRESS_ALIGNMENT	0x104B

Ejemplo:

```
void OnStart()
  {
//---
   int dCount= CLGetInfoInteger(0,CL_DEVICE_COUNT);
   for(int i = 0; i<dCount; i++)
     {
      int clCtx=CLContextCreate(i);
      if(clCtx == -1)
         Print("ERROR in CLContextCreate");
      string device;
      CLGetInfoString(clCtx,CL_DEVICE_NAME,device);
      Print(i,": ",device);
      uchar data[1024];
      uint size;
      CLGetDeviceInfo(clCtx,CL_DEVICE_VENDOR,data,size);
      Print("size = ",size);
      string str=CharArrayToString(data);
      Print(str);
     }
  }
//--- ejemplo de entradas en el historial Asesores Expertos
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      2: Advanced Micro Devices, Inc.
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 32
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Tahiti
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Intel(R) Corporation
```

© 2000-2025, MetaQuotes Ltd.

```
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 21
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      1:        Intel(R) Core(TM) i7-377
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      NVIDIA Corporation
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 19
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      0: GeForce GTX 580
```

© 2000-2025, MetaQuotes Ltd.

CLProgramCreate

Crea un programa OpenCL desde el código fuente.

```
int  CLProgramCreate(
   int           context,     // manejador para el contexto OpenCL
   const string  source       // código fuente
   );
```

La versión sobrecargada de la función crea un programa OpenCL y escribe el mensaje del compilador en la cadena pasada.

```
int  CLProgramCreate(
   int           context,     // manejador para el contexto OpenCL
   const string  source,      // código fuente
   string       &build_log    // cadena para recibir el log de compilación
   );
```

Parámetros

context

[in] Manejador del contexto OpenCL.

source

[in] Cadena con el código fuente OpenCL del programa.

&build_log

[in] Cadena para recibir los mensajes del compilador OpenCL.

Valor devuelto

Manejador para el objeto OpenCL en caso de ejecutarse con éxito. En caso del error devuelve -1. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

En estos momentos están previstos los siguientes códigos de errores:

- ERR_OPENCL_INVALID_HANDLE - un manejador inválido para context OpenCL,
- ERR_INVALID_PARAMETER - un parámetro literal inválido,
- ERR_NOT_ENOUGH_MEMORY - memoria insuficiente para completar la operación,
- ERR_OPENCL_PROGRAM_CREATE - un error interno de OpenCL o un error de compilación.

En algunas tarjetas gráficas, por defecto, está desactivado el modo de trabajo con los números del tipo double lo que provoca el error de compilación 5105. Para activar el modo del soporte de los números double, hay que insertar la siguiente directiva en el texto del programa OpenCL: #pragma OPENCL EXTENSION cl_khr_fp64 : enable. No obstante, si la tarjeta gráfica no soporta double, la inserción de esta directiva no servirá de nada.

Ejemplo:

```
//+------------------------------------------------------------------+
//| OpenCL kernel                                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
const string
cl_src=
       //--- by default some GPU doesn't support doubles
       //--- cl_khr_fp64 directive is used to enable work with doubles
       "#pragma OPENCL EXTENSION cl_khr_fp64 : enable      \r\n"
       //--- OpenCL kernel function
       "__kernel void Test_GPU(__global double *data,      \r\n"
       "                       const    int N,             \r\n"
       "                       const    int total_arrays)  \r\n"
       "  {                                                \r\n"
       "   uint kernel_index=get_global_id(0);             \r\n"
       "   if (kernel_index>total_arrays) return;          \r\n"
       "   uint local_start_offset=kernel_index*N;         \r\n"
       "   for(int i=0; i<N; i++)                          \r\n"
       "     {                                             \r\n"
       "       data[i+local_start_offset] *= 2.0;          \r\n"
       "     }                                             \r\n"
       "  }                                                \r\n";
//+------------------------------------------------------------------+
//| Test_CPU                                                         |
//+------------------------------------------------------------------+
bool Test_CPU(double &data[],const int N,const int id,const int total_arrays)
  {
//--- check array size
   if(ArraySize(data)==0) return(false);
//--- check array index
   if(id>total_arrays) return(false);
//--- calculate local offset for array with index id
   int local_start_offset=id*N;
//--- multiply elements by 2
   for(int i=0; i<N; i++)
     {
      data[i+local_start_offset]*=2.0;
     }
   return true;
  }
//---
#define ARRAY_SIZE   100  // size of the array
#define TOTAL_ARRAYS 5    // total arrays
//--- OpenCL handles
int cl_ctx;  // OpenCL context handle
int cl_prg;  // OpenCL program handle
int cl_krn;  // OpenCL kernel handle
int cl_mem;  // OpenCL buffer handle
//---
double DataArray1[]; // data array for CPU calculation
double DataArray2[]; // data array for GPU calculation
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnStart()
  {
//--- initialize OpenCL objects
//--- create OpenCL context
   if((cl_ctx=CLContextCreate())==INVALID_HANDLE)
     {
      Print("OpenCL not found. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL program
   if((cl_prg=CLProgramCreate(cl_ctx,cl_src))==INVALID_HANDLE)
     {
      CLContextFree(cl_ctx);
      Print("OpenCL program create failed. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL kernel
   if((cl_krn=CLKernelCreate(cl_prg,"Test_GPU"))==INVALID_HANDLE)
     {
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL kernel create failed. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL buffer
   if((cl_mem=CLBufferCreate(cl_ctx,ARRAY_SIZE*TOTAL_ARRAYS*sizeof(double),CL_MEM_REAI
     {
      CLKernelFree(cl_krn);
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL buffer create failed. Error=",GetLastError());
      return(1);
     }
//--- set OpenCL kernel constant parameters
   CLSetKernelArgMem(cl_krn,0,cl_mem);
   CLSetKernelArg(cl_krn,1,ARRAY_SIZE);
   CLSetKernelArg(cl_krn,2,TOTAL_ARRAYS);
//--- prepare data arrays
   ArrayResize(DataArray1,ARRAY_SIZE*TOTAL_ARRAYS);
   ArrayResize(DataArray2,ARRAY_SIZE*TOTAL_ARRAYS);
//--- fill arrays with data
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculate local start offset for jth array
      uint local_offset=j*ARRAY_SIZE;
      //--- prepare array with index j
      for(int i=0; i<ARRAY_SIZE; i++)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- fill arrays with function MathCos(i+j);
         DataArray1[i+local_offset]=MathCos(i+j);
         DataArray2[i+local_offset]=MathCos(i+j);
        }
     };
//--- test CPU calculation
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculation of the array with index j
      Test_CPU(DataArray1,ARRAY_SIZE,j,TOTAL_ARRAYS);
     }
//--- prepare CLExecute params
   uint  offset[]={0};
//--- global work size
   uint  work[]={TOTAL_ARRAYS};
//--- write data to OpenCL buffer
   CLBufferWrite(cl_mem,DataArray2);
//--- execute OpenCL kernel
   CLExecute(cl_krn,1,offset,work);
//--- read data from OpenCL buffer
   CLBufferRead(cl_mem,DataArray2);
//--- total error
   double total_error=0;
//--- compare results and calculate error
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculate local offset for jth array
      uint local_offset=j*ARRAY_SIZE;
      //--- compare the results
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         double v1=DataArray1[i+local_offset];
         double v2=DataArray2[i+local_offset];
         double delta=MathAbs(v2-v1);
         total_error+=delta;
         //--- show first and last arrays
         if((j==0) || (j==TOTAL_ARRAYS-1))
            PrintFormat("array %d of %d, element [%d]:  %f, %f, [error]=%f",j+1,TOTAL_
        }
     }
   PrintFormat("Total error: %f",total_error);
//--- delete OpenCL objects
//--- free OpenCL buffer
   CLBufferFree(cl_mem);
//--- free OpenCL kernel
   CLKernelFree(cl_krn);
//--- free OpenCL program
   CLProgramFree(cl_prg);
//--- free OpenCL context
```

© 2000-2025, MetaQuotes Ltd.

```
   CLContextFree(cl_ctx);
//---
   return(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

CLProgramFree

Elimina un programa OpenCL.

```
void  CLProgramFree(
   int  program     // manejador para el objeto OpenCL
   );
```

Parámetros

program

[in] Manejador del objeto OpenCL.

Valor devuelto

Ninguno. En caso de un error interno, se cambia el valor de _LastError. Para obtener la información sobre el error, llame a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLKernelCreate

Crea un punto de acceso al programa OpenCL y devuelve el manejador para él.

```
int  CLKernelCreate(
   int           program,        // manejador para el objeto OpenCL
   const string  kernel_name     // nombre del kernel
   );
```

Parámetros

program

[in] Manejador para un objeto del programa OpenCL.

kernel_name

[in] Nombre de la función kernel, es decir el nombre del punto de acceso en el programa OpenCL correspondiente a partir del cual se empieza la ejecución.

Valor devuelto

Manejador para el objeto OpenCL en caso de ejecutarse con éxito. En caso del error devuelve -1. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

En estos momentos están previstos los siguientes códigos de errores:

- ERR_OPENCL_INVALID_HANDLE - manejador inválido para *program* OpenCL,
- ERR_INVALID_PARAMETER - un parámetro literal inválido,
- ERR_OPENCL_TOO_LONG_KERNEL_NAME - nombre del kernel contiene más de 127 caracteres,
- ERR_OPENCL_KERNEL_CREATE - un error interno a la hora de crear el objeto OpenCL.

© 2000-2025, MetaQuotes Ltd.

CLKernelFree

Elimina la función del arranque OpenCL.

```
void  CLKernelFree(
   int  kernel      // manejador para el kernel del programa OpenCL
   );
```

Parámetros

kernel_name

[in] Manejador del objeto kernel.

Valor devuelto

Ninguno. En caso de un error interno, se cambia el valor de _LastError. Para obtener la información sobre el error, llame a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLSetKernelArg

Establece un parámetro para la función OpenCL.

```
bool  CLSetKernelArg(
   int   kernel,        // manejador para el kernel del programa OpenCL
   uint  arg_index,     // número del argumento de la función OpenCL
   void  arg_value      // código fuente
   );
```

Parámetros

kernel

[in] Manejador para el kernel del programa OpenCL.

arg_index

[in] El número del argumento de la función, la numeración se comienza desde cero.

arg_value

[in] Valor del argumento de la función.

Valor devuelto

En caso del éxito la función devuelve true, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

En estos momentos están previstos los siguientes códigos de errores:

- ERR_INVALID_PARAMETER,
- ERR_OPENCL_INVALID_HANDLE - manejador inválido para el kernel OpenCL,
- ERR_OPENCL_SET_KERNEL_PARAMETER - error interno de OpenCL.

© 2000-2025, MetaQuotes Ltd.

CLSetKernelArgMem

Establece el búfer OpenCL como parámetro de la función OpenCL.

```
bool  CLSetKernelArgMem(
   int   kernel,           // manejador para el kernel del programa OpenCL
   uint  arg_index,        // número del argumento de la función OpenCL
   int   cl_mem_handle     // manejador del búfer OpenCL
   );
```

Parámetros

kernel

[in] Manejador para el kernel del programa OpenCL.

arg_index

[in] El número del argumento de la función, la numeración se comienza desde cero.

cl_mem_handle

[in] Manejador para el búfer OpenCL.

Valor devuelto

En caso del éxito la función devuelve true, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLSetKernelArgMemLocal

Define el búfer local como argumento de la función núcleo.

```
bool  CLSetKernelArgMemLocal(
   int    kernel,           // manejador para el núcleo del programa OpenCL
   uint   arg_index,        // número del argumento OpenCL de función
   ulong  local_mem_size    // tamaño del búfer
   );
```

Parámetros

kernel

[in] Manejador para el núcleo del programa OpenCL.

arg_index

[in] Número del argumento de la función, la numeración se comienza desde cero.

local_mem_size

[in] Tamaño del búfer en bytes.

Valor devuelto

En caso del éxito la función devuelve true, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLBufferCreate

Crea un búfer OpenCL y devuelve su manejador.

```
int  CLBufferCreate(
   int   context,     // manejador para el contexto OpenCL
   uint  size,        // tamaño del búfer
   uint  flags        // combinación de banderas que establecen las propiedades del bú
   );
```

Parámetros

context

[in] Manejador para context OpenCL.

size

[in] Tamaño del búfer en bytes.

flags

[in] Las propiedades del búfer que se establecen mediante la combinación de banderas: CL_MEM_READ_WRITE, CL_MEM_WRITE_ONLY, CL_MEM_READ_ONLY, CL_MEM_ALLOC_HOST_PTR.

Valor devuelto

Manejador para el búfer OpenCL en caso de ejecutarse con éxito. En caso del error devuelve -1. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

En estos momentos están previstos los siguientes códigos de errores:

- ERR_OPENCL_INVALID_HANDLE - un manejador inválido para el contexto OpenCL,
- ERR_NOT_ENOUGH_MEMORY - memoria insuficiente,
- ERR_OPENCL_BUFFER_CREATE - un error interno de creación del búfer.

© 2000-2025, MetaQuotes Ltd.

CLBufferFree

Elimina el búfer OpenCL.

```
void  CLBufferFree(
   int   buffer     // manejador para el búfer OpenCL
   );
```

Parámetros

buffer

[in] Manejador para el búfer OpenCL.

Valor devuelto

Ninguno. En caso de un error interno, se cambia el valor de _LastError. Para obtener la información sobre el error, llame a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLBufferWrite

Escribe un array en el búfer OpenCL y devuelve el número de elementos escritos.

```
uint  CLBufferWrite(
   int           buffer,                    // manejador para el búfer OpenCL
   const void&   data[],                    // array de valores
   uint          buffer_offset=0,           // offset en el búfer OpenCL en bytes, por
   uint          data_offset=0,             // offset dentro del array en elementos, po
   uint          data_count=WHOLE_ARRAY     // número de valores del array para la esc
   );
```

Asimismo, existe una versión para trabajar con matrices y vectores.

Escribe los valores de una matriz en un búfer y retorna true en caso de éxito.

```
uint  CLBufferWrite(
   int           buffer,                    // manejador al búfer OpenCL
   uint          buffer_offset,             // desplazamiento en el búfer OpenCL en by
   matrix<T>     &mat                       // matriz de valores para anotar en el búf
   );
```

Escribe los valores de un vector en un búfer y retorna true en caso de éxito.

```
uint  CLBufferWrite(
   int           buffer,                    // manejador al búfer OpenCL
   uint          buffer_offset,             // desplazamiento en el búfer OpenCL en by
   vector<T>     &vec                       // vector de valores para anotar en el búf
   );
```

Parámetros

buffer

[in] Manejador del búfer OpenCL.

data[]

[in] Array de valores que hay que escribir en el búfer OpenCL. Se pasa por referencia.

buffer_offset

[in] Desplazamiento (offset) en el búfer OpenCL en bytes, a partir del cual se empieza la escritura. Por defecto, la escritura se hace desde el principio del búfer.

data_offset

[in] Índice del primer elemento del array a partir del cual se cogen los valores desde el array para la escritura en el búfer OpenCL. Por defecto, los valores se cogen desde el principio del array.

data_count

[in] Número de valores que hay que escribir. Por defecto, todos los valores del array.

mat

[out] La matriz para leer los datos del búfer puede ser de cualquiera de los tres tipos — matrix, matrixf o matrixc.

© 2000-2025, MetaQuotes Ltd.

vec

[out] El vector para leer los datos del búfer puede ser de cualquiera de los tres tipos — vector, vectorf o vectorc.

Valor devuelto

Número de valores que se han escrito. En caso del error, devuelve 0. Para obtener la información sobre el error, utilice la función GetLastError().

true en caso de ejecución exitosa al trabajar con una matriz o vector; en caso de error, se retornará false.

Nota

Para los arrays unidimensionales, el número del elemento a partir del que se empieza la lectura de dato para su escritura en el búfer OpenCL, se calcula teniendo en cuenta la bandera AS_SERIES.

Un array con dos o más dimensiones se representa como unidimensional. En este caso *data_offset* es el número de elementos que hay que saltar en la representación, y no el número de elementos en la primera dimensión.

Ejemplo de multiplicación de matrices usando el método MatMul y cálculos paralelos en OpenCL

```
#define M       3000      // número de líneas en la primera matriz
#define K       2000      // el número de columnas en la primera matriz es igual a núr
#define N       3000      // número de columnas en la segunda matriz

//+------------------------------------------------------------------+
const string clSrc=
  "#define N     "+IntegerToString(N)+"                              \r\n"
  "#define K     "+IntegerToString(K)+"                              \r\n"
  "                                                                 \r\n"
  "__kernel void matricesMul( __global float *in1,                  \r\n"
  "                           __global float *in2,                  \r\n"
  "                           __global float *out  )                \r\n"
  "{                                                                \r\n"
  "  int m = get_global_id( 0 );                                    \r\n"
  "  int n = get_global_id( 1 );                                    \r\n"
  "  float sum = 0.0;                                               \r\n"
  "  for( int k = 0; k < K; k ++ )                                  \r\n"
  "     sum += in1[ m * K + k ] * in2[ k * N + n ];                 \r\n"
  "  out[ m * N + n ] = sum;                                        \r\n"
  "}                                                                \r\n";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- inicializamos el generador de números aleatorios
  MathSrand((int)TimeCurrent());
```

© 2000-2025, MetaQuotes Ltd.

```
//--- rellenamos la matriz del tamaño indicado con valores aleatorios
  matrixf mat1(M, K, MatrixRandom) ;     // primera matriz
  matrixf mat2(K, N, MatrixRandom);      // segunda matriz

//--- calculamos el producto de las matrices de forma ingenua
  uint start=GetTickCount();
  matrixf matrix_naive=matrixf::Zeros(M, N);// aquí anotamos el resultado de la multip
  for(int m=0; m<M; m++)
    for(int k=0; k<K; k++)
      for(int n=0; n<N; n++)
        matrix_naive[m][n]+=mat1[m][k]*mat2[k][n];
  uint time_naive=GetTickCount()-start;

//--- calculamos el producto de las matrices usando MatMull
  start=GetTickCount();
  matrixf matrix_matmul=mat1.MatMul(mat2);
  uint time_matmul=GetTickCount()-start;

//--- calculamos el producto de las matrices en OpenCL
  matrixf matrix_opencl=matrixf::Zeros(M, N);
  int cl_ctx;              // manejador de contexto
  if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL no ha sido encontrado, salimos");
    return;
   }
  int cl_prg;              // manejador del programa
  int cl_krn;              // manejador del kernel
  int cl_mem_in1;          // manejador del primer búfer (de entrada)
  int cl_mem_in2;          // manejador del segundo búfer (de entrada)
  int cl_mem_out;          // manejador del tercer búfer (de entrada)
//--- creamos el programa y el kernel
  cl_prg = CLProgramCreate(cl_ctx, clSrc);
  cl_krn = CLKernelCreate(cl_prg, "matricesMul");
//--- creamos los tres búferes para las tres matrices
  cl_mem_in1=CLBufferCreate(cl_ctx, M*K*sizeof(float), CL_MEM_READ_WRITE);
  cl_mem_in2=CLBufferCreate(cl_ctx, K*N*sizeof(float), CL_MEM_READ_WRITE);
//--- la tercera matriz es de salida
  cl_mem_out=CLBufferCreate(cl_ctx, M*N*sizeof(float), CL_MEM_READ_WRITE);
//--- establecemos los argumentos del kernel
  CLSetKernelArgMem(cl_krn, 0, cl_mem_in1);
  CLSetKernelArgMem(cl_krn, 1, cl_mem_in2);
  CLSetKernelArgMem(cl_krn, 2, cl_mem_out);
//--- escribimos las matrices en los búferes del dispositivo
  CLBufferWrite(cl_mem_in1, 0, mat1);
  CLBufferWrite(cl_mem_in2, 0, mat2);
  CLBufferWrite(cl_mem_out, 0, matrix_opencl);
//--- iniciando el tiempo de ejecución del código de OpenCL
  start=GetTickCount();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- establecemos los parámetros del espacio de trabajo de la tarea y ejecutamos el p
  uint  offs[2] = {0, 0};
  uint works[2] = {M, N};
  start=GetTickCount();
  bool ex=CLExecute(cl_krn, 2, offs, works);
//--- calculamos el resultado en la matriz
  if(CLBufferRead(cl_mem_out, 0, matrix_opencl))
    PrintFormat("Leída la matriz [%d x %d]: ", matrix_opencl.Rows(), matrix_opencl.Col
   else
      Print("CLBufferRead(cl_mem_out, 0, matrix_opencl failed. Error ",GetLastError())
  uint time_opencl=GetTickCount()-start;
  Print("Comparamos el tiempo de cálculo con cada método");
  PrintFormat("Naive product time = %d ms",time_naive);
  PrintFormat("MatMul product time = %d ms",time_matmul);
  PrintFormat("OpenCl product time = %d ms",time_opencl);
//--- liberamos todos los contextos de OpenCL
  CLFreeAll(cl_ctx, cl_prg, cl_krn, cl_mem_in1, cl_mem_in2, cl_mem_out);

//--- comparamos entre sí todas las matrices de resultados obtenidas
  Print("¿Cuántos errores de divergencia hay entre las matrices de resultados?");
  ulong errors=matrix_naive.Compare(matrix_matmul,(float)1e-12);
  Print("matrix_direct.Compare(matrix_matmul,1e-12)=",errors);
  errors=matrix_matmul.Compare(matrix_opencl,float(1e-12));
  Print("matrix_matmul.Compare(matrix_opencl,1e-12)=",errors);
/*
  Resultado:

   Leída la matriz [3000 x 3000]:
   Comparamos el tiempo de cálculo con cada método
   Naive product time = 54750 ms
   MatMul product time = 4578 ms
   OpenCl product time = 922 ms
   ¿Cuántos errores de divergencia hay entre las matrices de resultados?
   matrix_direct.Compare(matrix_matmul,1e-12)=0
   matrix_matmul.Compare(matrix_opencl,1e-12)=0
*/
 }
//+------------------------------------------------------------------+
//| Rellena la matriz con valores aleatorios                         |
//+------------------------------------------------------------------+
void MatrixRandom(matrixf& m)
 {
  for(ulong r=0; r<m.Rows(); r++)
   {
    for(ulong c=0; c<m.Cols(); c++)
     {
      m[r][c]=(float)((MathRand()-16383.5)/32767.);
     }
   }
```

© 2000-2025, MetaQuotes Ltd.

```
 }
//+------------------------------------------------------------------+
//| Liberamos todos los contextos de OpenCL                          |
//+------------------------------------------------------------------+
void CLFreeAll(int cl_ctx, int cl_prg, int cl_krn,
               int cl_mem_in1, int cl_mem_in2, int cl_mem_out)
 {
//--- eliminamos en una secuencia inversa todos los contextos de OpenCL creados
  CLBufferFree(cl_mem_in1);
  CLBufferFree(cl_mem_in2);
  CLBufferFree(cl_mem_out);
  CLKernelFree(cl_krn);
  CLProgramFree(cl_prg);
  CLContextFree(cl_ctx);
 }
```

© 2000-2025, MetaQuotes Ltd.

CLBufferRead

Lee el búfer OpenCL en un array y devuelve el número de elementos leídos.

```
uint  CLBufferRead(
   int           buffer,                  // manejador para el búfer OpenCL
   const void&   data[],                  // array de valores
   uint          buffer_offset=0,         // offset en el búfer OpenCL en bytes, por
   uint          data_offset=0,           // offset dentro del array en elementos, p
   uint          data_count=WHOLE_ARRAY   // número de valores del búfer para la lect
   );
```

Asimismo, existe una versión para trabajar con matrices y vectores.

Lee el búfer OpenCL en una matriz y retorna true en caso de éxito.

```
uint  CLBufferRead(
   int           buffer,                  // manejador al búfer OpenCL
   uint          buffer_offset,           // desplazamiento en el búfer OpenCL en by
   const matrxi& mat,                     // matriz para obtener los valores  del bú
   ulong         rows=-1,                 // número de líneas en la matriz
   ulong         cols=-1                  // número de columnas en la matriz
   );
```

Lee el búfer OpenCL en el vector y retorna true en caso de éxito.

```
uint  CLBufferRead(
   int           buffer,                  // manejador al búfer OpenCL
   uint          buffer_offset,           // desplazamiento en el búfer OpenCL en by
   const vector& vec,                     // vector para obtener los valores de la b
   ulong         size-1,                  // longitud del vector
   );
```

Parámetros

buffer

[in] Manejador del búfer OpenCL.

data[]

[in] Array para recibir valores desde el búfer OpenCL. Se pasa por referencia.

buffer_offset

[in] Desplazamiento (offset) en el búfer OpenCL en bytes, a partir del cual se empieza la lectura. Por defecto, la lectura se empieza desde el principio del búfer.

data_offset

[in] Índice del primer elemento del array para la escritura de los valores del búfer OpenCL. Por defecto, los valores leídos se escriben en el array desde el índice cero.

data_count

[in] Número de valores que hay que leer. Por defecto, se lee el búfer OpenCL entero.

mat

© 2000-2025, MetaQuotes Ltd.

[out] La matriz para leer los datos del búfer puede ser de cualquiera de los tres tipos — matrix, matrixf o matrixc.

vec

[out] El vector para leer los datos del búfer puede ser de cualquiera de los tres tipos — vector, vectorf o vectorc.

rows=-1

[in] Si el parámetro ha sido indicado, también se deberá especificar el parámetro *cols*. Si no se especifican las nuevas dimensiones de la matriz, se utilizarán las dimensiones actuales. Si el valor es -1, el número de filas no cambiará.

cols=-1

[in] Si el parámetro no ha sido indicado, el parámetro *rows* también deberá omitirse. Para las matrices, se aplica la regla siguiente: o bien se indican ambos parámetros, o bien ninguno, de lo contrario se producirá un error. Si se indican ambos parámetros (*rows y cols*), el tamaño de la matriz será modificado. Si el valor es igual a -1, el número de columnas no cambiará.

size=-1

[in] Si el parámetro no ha sido indicado o el valor es -1, la longitud del vector no cambiará.

Valor devuelto

Número de valores leídos. En caso del error, devuelve 0. Para obtener la información sobre el error, utilice la función GetLastError().

true en caso de ejecución exitosa al trabajar con una matriz o vector; en caso de error, se retornará false.

Nota

Para los arrays unidimensionales, el número del elemento en el que se empieza la escritura de datos desde el búfer OpenCL, se calcula teniendo en cuenta la bandera AS_SERIES.

Un array con dos o más dimensiones se representa como unidimensional. En este caso *data_offset* es el número de elementos que hay que saltar en la representación, y no el número de elementos en la primera dimensión.

Ejemplo de cálculo del número Pi según la fórmula:

$$\pi = \lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N-1}\frac{4}{1+\left(\frac{2k+1}{2N}\right)^2} = 16\lim_{N\to\infty}N\sum_{k=0}^{N-1}\frac{1}{4N^2+(2k+1)^2}$$

```
#define  _num_steps        1000000000
#define  _divisor          40000
#define  _step             1.0 / _num_steps
#define  _intrnCnt         _num_steps / _divisor
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
string D2S(double arg, int digits) { return DoubleToString(arg, digits); }
string I2S(int arg)                { return IntegerToString(arg); }

//--- OpenCL programm code
const string clSource=
  "#define _step "+D2S(_step, 12)+"                    \r\n"
  "#define _intrnCnt "+I2S(_intrnCnt)+"                \r\n"
  "                                                    \r\n"
  "__kernel void Pi( __global double *out )            \r\n"
  "{                                                   \r\n"
  "  int i = get_global_id( 0 );                       \r\n"
  "  double partsum = 0.0;                             \r\n"
  "  double x = 0.0;                                   \r\n"
  "  long from = i * _intrnCnt;                        \r\n"
  "  long to = from + _intrnCnt;                       \r\n"
  "  for( long j = from; j < to; j ++ )                \r\n"
  "  {                                                 \r\n"
  "     x = ( j + 0.5 ) * _step;                       \r\n"
  "     partsum += 4.0 / ( 1. + x * x );               \r\n"
  "  }                                                 \r\n"
  "  out[ i ] = partsum;                               \r\n"
  "}                                                   \r\n";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart()
 {
  Print("Pi Calculation: step = "+D2S(_step, 12)+"; _intrnCnt = "+I2S(_intrnCnt));
//--- prepare OpenCL contexts
  int clCtx;
  if((clCtx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL not found");
    return(-1);
   }
  int clPrg = CLProgramCreate(clCtx, clSource);
  int clKrn = CLKernelCreate(clPrg, "Pi");
  int clMem=CLBufferCreate(clCtx, _divisor*sizeof(double), CL_MEM_READ_WRITE);
  CLSetKernelArgMem(clKrn, 0, clMem);

  const uint offs[1]  = {0};
  const uint works[1] = {_divisor};
//--- launch OpenCL program
  ulong start=GetMicrosecondCount();
  if(!CLExecute(clKrn, 1, offs, works))
   {
    Print("CLExecute(clKrn, 1, offs, works) failed! Error ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
    CLFreeAll(clMem, clKrn, clPrg, clCtx);
    return(-1);
   }
//--- get results from OpenCL device
  vector buffer(_divisor);
  if(!CLBufferRead(clMem, 0, buffer))
   {
    Print("CLBufferRead(clMem, 0, buffer) failed! Error ", GetLastError());
    CLFreeAll(clMem, clKrn, clPrg, clCtx);
    return(-1);
   }
//--- sum all values to calculate Pi
  double Pi=buffer.Sum()*_step;

  double time=(GetMicrosecondCount()-start)/1000.;
  Print("OpenCL: Pi calculated for "+D2S(time, 2)+" ms");
  Print("Pi = "+DoubleToString(Pi, 12));
//--- free memory
  CLFreeAll(clMem, clKrn, clPrg, clCtx);
//--- success
  return(0);
 }
  /*
  Pi Calculation: step = 0.000000001000; _intrnCnt = 25000
  OpenCL: GPU device 'Ellesmere' selected
  OpenCL: Pi calculated for 99.98 ms
  Pi = 3.141592653590
  */
//+------------------------------------------------------------------+
//| Auxiliary routine to free memory                                 |
//+------------------------------------------------------------------+
void CLFreeAll(const int clMem, const int clKrn, const int clPrg, const int clCtx)
 {
  CLBufferFree(clMem);
  CLKernelFree(clKrn);
  CLProgramFree(clPrg);
  CLContextFree(clCtx);
 }
```

© 2000-2025, MetaQuotes Ltd.

CLExecute

Ejecuta un programa OpenCL. Hay 3 variantes de esta función:

1. Inicio de la función kernel usando un solo núcleo

```
bool  CLExecute(
   int           kernel                  // manejador para el kernel de un programa C
   );
```

2. Inicio de varias copias kernel (función OpenCL) con la descripción del espacio de tareas

```
bool  CLExecute(
   int           kernel,                 // manejador para el kernel del programa Ope
   uint          work_dim,               // dimensión del espacio de tareas
   const uint&   global_work_offset[],   // offset inicial en el espacio de tareas
   const uint&   global_work_size[]      // número total de tareas
   );
```

3. Inicio de varias copias kernel (función OpenCL) con la descripción del espacio de tareas y especificación del tamaño del subconjunto local de tareas en grupo

```
bool  CLExecute(
   int           kernel,                 // manejador para el kernel del programa Ope
   uint          work_dim,               // dimensión del espacio de tareas
   const uint&   global_work_offset[],   // offset inicial en el espacio de tareas
   const uint&   global_work_size[],     // número total de tareas
   const uint&   local_work_size[]       // número de tareas en el grupo local
   );
```

Parámetros

kernel

[in] Manejador para el kernel OpenCL.

work_dim

[in] Dimensión del espacio de tareas.

global_work_offset[]

[in] Desplazamiento inicial en el espacio de tareas.

global_work_size[]

[in] Tamaño del subconjunto de tareas.

local_work_size[]

[in] Tamaño del subconjunto local de tareas en el grupo.

Valor devuelto

En caso del éxito la función devuelve true, de lo contrario devuelve false. Para obtener la información sobre el error, hay que llamar a la función GetLastError().

Nota

© 2000-2025, MetaQuotes Ltd.

Vamos a ver el uso de los parámetros con la ayuda de los siguientes ejemplos:

- *work_dim* establece la dimensión del array *work_items[]* que describe las tareas. Si *work_dim*=3, se usa el array de tres dimensiones *work_items[N1, N2, N3]*.
- *global_work_size[]* contiene los valores que establecen el tamaño del array *work_items[]*. Si tenemos work_dim=3, y en consecuencia, el array *global_work_size[3]* puede ser {40, 100, 320}. Entonces tenemos *work_items[40, 100, 320]*. Eso significa que el número total de tareas es igual a 40 x 100 x 320 = 1 280 000.
- *local_work_size[]* establece el conjunto de tareas que van a ejecutarse por el kernel especificado del programa OpenCL. Su dimensión es igual a la dimensión *work_items[]* y permite dividir el subconjunto general de las tareas en los subconjuntos más pequeños sin restas de la división. Prácticamente, los tamaños del array *local_work_size[]* tiene que ser seleccionado de tal manera que el conjunto global de tareas *work_items[]* se divida en los subconjuntos del menor tamaño. En este ejemplo queda bien *local_work_size[3]={10, 10, 10}*, ya que *work_items[40, 100, 320]* se puede reunir sin la resta desde el array *local_items[10, 10, 10]*.

© 2000-2025, MetaQuotes Ltd.

CLExecutionStatus

Retorna el estado de ejecución del programa OpenCL.

```
int  CLExecutionStatus(
   int   kernel            // manejador para el núcleo del programa OpenCL
   );
```

Parámetros

kernel

[in] Manejador para el núcleo del programa OpenCL.

Valor devuelto

Retorna el estado del programa OpenCL, el valor puede ser uno de los siguientes:

- CL_COMPLETE=0 - programa finalizado,
- CL_RUNNING=1 - programa en ejecución,
- CL_SUBMITTED=2 - programa enviado para su ejecución,
- CL_QUEUED=3 - el programa se halla en la cola para su ejecución,
- -1 (menos uno) - ha sucedido un error al ejecutar CLExecutionStatus().

© 2000-2025, MetaQuotes Ltd.

Trabajo con la base de datos

Las funciones para trabajar con las bases de datos usan el popular y sencillo motor SQLite. La comodidad de este motor reside en que toda la base de datos se encuentra en un único archivo en el disco duro de la computadora del usuario.

Con la ayuda de estas funciones, resulta muy sencillo crear recuadros, añadir datos a estos, realizar modificaciones y generar muestras con solicitudes SQL simples:

- obtener la historia comercial y las cotizaciones en cualquier formato,
- guardar los resultados de la optimización y la simulación,
- preparar e intercambiar datos con otros paquetes analíticos,
- guardar los ajustes y estados de los programas MQL5.

Además, en las solicitudes se pueden utilizar funciones estadísticas y matemáticas.

Las funciones para trabajar con las bases de datos permiten sustituir las operaciones más repetidas en cuanto al procesamiento de grandes matrices de datos por solicitudes SQL, lo cual posibilita en muchos casos, en lugar de programar complejos ciclos y comparaciones, usar las llamadas de DatabaseExecute/DatabasePrepare. Para obtener fácilmente los resultados de la solicitud en una estructura ya preparada, use la función DatabaseReadBind, que permite leer directamente todos los campos de la entrada en una sola llamada.

Para acelerar las operaciones de lectura, registro y modificación de la base de datos, es posible abrir/crear en la memoria operativa con la bandera DATABASE_OPEN_MEMORY, pero, en este caso, la base estará disponible en un programa concreto y no se dividirá con ningún otro. Al trabajar con las bases de datos que se encuentran en el disco duro, es necesario envolver las inserciones/cambios masivos de datos en las transacciones con la ayuda de DatabaseTransactionBegin/DatabaseTransactionCommit/DatabaseTransactionRollback, lo cual ofrece una velocidad cientos de veces superior.

Para empezar a trabajar con estas funciones, basta con leer el artículo SQLite: trabajo nativo con bases de datos en SQL en MQL5.

Función	Acción
DatabaseOpen	Abre o crea una base de datos en el archivo indicado
DatabaseClose	Cierra una base de datos
DatabaseImport	Importa a un recuadro los datos de un archivo
DatabaseExport	Exporta un recuadro o el resultado de la ejecución de una solicitud SQL a un archivo CSV
DatabasePrint	Imprime el recuadro o resultado de la ejecución de una solicitud SQL en el diario de expertos
DatabaseTableExists	Comprueba la presencia de un recuadro en la base de datos
DatabaseExecute	Ejecuta una solicitud a la base de datos indicada
DatabasePrepare	Crea el manejador de una solicitud, que después puede ser ejecutada con la ayuda de DatabaseRead()

© 2000-2025, MetaQuotes Ltd.

Función	Acción
DatabaseReset	Resetea la solicitud a su estado inicial, igual que tras la llamada de DatabasePrepare()
DatabaseBind	Establece el valor de un parámetro en la solicitud
DatabaseBindArray	Establece una matriz como valor del parámetro.
DatabaseRead	Ejecuta el paso a la siguiente entrada en el resultado de la solicitud
DatabaseReadBind	Pasa a la siguiente entrada y lee los datos en la estructura de esta
DatabaseFinalize	Elimina la solicitud creada en DatabasePrepare()
DatabaseTransactionBegin	Comienza la ejecución de una transacción
DatabaseTransactionCommit	Finaliza la ejecución de una transacción
DatabaseTransactionRollback	Ejecuta el retroceso de una transacción
DatabaseColumnsCount	Obtiene el número de campos en una solicitud
DatabaseColumnName	Obtiene el nombre de un campo según el número
DatabaseColumnType	Obtiene el tipo de un campo según el número
DatabaseColumnSize	Obtiene el tamaño del campo en bytes
DatabaseColumnText	Obtiene de la entrada actual el valor del campo en forma de línea
DatabaseColumnInteger	Obtiene de la entrada actual un valor del tipo int
DatabaseColumnLong	Obtiene de la entrada actual un valor del tipo long
DatabaseColumnDouble	Obtiene de la entrada actual un valor del tipo double
DatabaseColumnBlob	Obtiene de la entrada actual el valor del campo en forma de matriz

Funciones estadísticas:

- mode — moda
- median — mediana (percentil 50)
- percentile_25 — percentil 25
- percentile_75 — percentil 75
- percentile_90 — percentil 90
- percentile_95 — percentil 95
- percentile_99 — percentil 99
- stddev o stddev_samp — desviación estándar de la muestra
- stddev_pop — desviación estándar de la población
- variance or var_samp — varianza de la muestra

© 2000-2025, MetaQuotes Ltd.

- var_pop — varianza de la población

Funciones matemáticas

- acos(X) - arcocoseno en radianes
- acosh(X) - arcocoseno hiperbólico
- asin(X) - arcoseno en radianes
- asinh(X) - arcoseno hiperbólico
- atan(X) - arcotangente en radianes
- atan2(X,Y) - arcotangente en radianes de la relación X/Y
- atanh(X) - arcotangente hiperbólico
- ceil(X) - redondeo hasta el número entero superior
- ceiling(X) - redondeo hasta el número entero superior
- cos(X) - coseno del ángulo en radianes
- cosh(X) - coseno hiperbólico
- degrees(X) - convierte los radianes en un ángulo
- exp(X) - función exponencial
- floor(X) - redondeo hasta el número entero inferior
- ln(X) - logaritmo natural
- log(B,X) - logaritmo de base indicada
- log(X) - logaritmo decimal
- log10(X) - logaritmo decimal
- log2(X) - logaritmo de base 2
- mod(X,Y) - remanente de la división
- pi() - número Pi aproximado
- pow(X,Y) - potencia de base indicada
- power(X,Y) - potencia de base indicada
- radians(X) - convierte un ángulo en radianes
- sin(X) - seno de un ángulo en radianes
- sinh(X) - seno hiperbólico
- sqrt(X) - raíz cuadrada
- tan(X) - tangente de un ángulo en radianes
- tanh(X) - tangente hiperbólica
- trunc(X) - recorta hasta el número entero más próximo a 0

Ejemplo:

```
select
  count(*) as book_count,
  cast(avg(parent) as integer) as mean,
  cast(median(parent) as integer) as median,
  mode(parent) as mode,
  percentile_90(parent) as p90,
```

© 2000-2025, MetaQuotes Ltd.

```
  percentile_95(parent) as p95,
  percentile_99(parent) as p99
from moz_bookmarks;
```

© 2000-2025, MetaQuotes Ltd.

DatabaseOpen

Abre o crea una base de datos en el archivo indicado.

```
int  DatabaseOpen(
   string  filename,       // nombre del archivo
   uint    flags,          // combinación de banderas
   );
```

Parámetros

filename

[in] Nombre del archivo respecto a la carpeta "MQL5\Files".

flags

[in] Combinación de banderas de la enumeración ENUM_DATABASE_OPEN_FLAGS.

Valor retornado

Si la ejecución tiene éxito, la función retorna el manejador de la base de datos, que después se usará para acceder a los datos de la base; en el caso contrario, retorna INVALID_HANDLE. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INTERNAL_ERROR (4001) - error crítico del sistema de ejecución;
- ERR_WRONG_INTERNAL_PARAMETER (4002) - error interno de acceso a la carpeta "MQL5\Files";
- ERR_INVALID_PARAMETER (4003) - la ruta al archivo de la base de datos contiene una línea vacía o se ha establecido una combinación de banderas incompatible;
- ERR_NOT_ENOUGH_MEMORY (4004) - memoria insuficiente;
- ERR_WRONG_FILENAME (5002) - el nombre del archivo de la base de datos es incorrecto;
- ERR_TOO_LONG_FILENAME (5003) - la ruta absoluta al archivo de la base de datos ha superado la longitud máxima;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - se ha superado el número máximo permitido de objetos Database;
- ERR_DATABASE_CONNECT (5123) - error al conectarse a la base de datos;
- ERR_DATABASE_MISUSE (5621) - uso incorrecto de la biblioteca SQLite.

Observación

Si en el parámetro *filename* se indica NULL o una línea vacía "", en el disco se creará un archivo temporal que será automáticamente eliminado tras cerrar la conexión con la base de datos.

Si en el parámetro *filename* se indica ":memory:", la base de datos se creará en la memoria; en este caso, además, la base de datos será automáticamente eliminada tras cerrar la conexión con ella.

Si en el parámetro *flags* no se indica ninguna de las banderas DATABASE_OPEN_READONLY o DATABASE_OPEN_READWRITE, se usará la bandera DATABASE_OPEN_READWRITE.

Si no se ha establecido una extensión para el archivo, se usará la extensión ".sqlite"

ENUM_DATABASE_OPEN_FLAGS

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
DATABASE_OPEN_READONLY	Abrir solo para lectura
DATABASE_OPEN_READWRITE	Abrir solo para lectura y grabación
DATABASE_OPEN_CREATE	Crear un archivo en el disco, si aún no existe
DATABASE_OPEN_MEMORY	Crear una base de datos en la memoria operativa
DATABASE_OPEN_COMMON	El archivo se encuentra en la carpeta general de todos los terminales

Ver también

DatabaseClose

© 2000-2025, MetaQuotes Ltd.

DatabaseClose

Cierra una base de datos.

```
void  DatabaseClose(
   int  database      // manejador de base de datos recibido en DatabaseOpen
   );
```

Parámetros

database

[in] Manejador de base de datos recibido en DatabaseOpen().

Valor retornado

No.

Observación

Después de llamar a DatabaseClose, todos los manejadores de las solicitudes a la base de datos son eliminados y quedan invalidados.

Si el manejador no es válido, la función mostrará el error ERR_DATABASE_INVALID_HANDLE. Es posible comprobar el error con la ayuda de GetLastError().

Ver también

DatabaseOpen, DatabasePrepare

© 2000-2025, MetaQuotes Ltd.

DatabaseImport

Importa a un recuadro los datos de un archivo.

```
long  DatabaseImport(
   int           database,          // manejador de base de datos recibido en Database
   const string  table,             // nombre del recuadro para insertar los datos
   const string  filename,          // nombre del archivo para importar los datos
   uint          flags,             // combinación de banderas
   const string  separator,         // separador de datos
   ulong         skip_rows,         // cuantas primeras líneas se omiten
   const string  skip_comments      // línea de símbolos que determinan los comentario
   );
```

Parámetros

database

[in] Manejador de base de datos recibido en DatabaseOpen().

table

[in] Nombre del recuadro al que se añadirán los datos del archivo.

filename

[in] Archivo CSV o archivo ZIP para leer los datos, el nombre puede contener subcarpetas y se establece con respecto a la carpeta MQL5\Files.

flags

[in] Combinación de banderas de la enumeración ENUM_DATABASE_IMPORT_FLAGS.

separator

[in] Separador de datos en el archivo CSV.

skip_rows

[in] Número de líneas iniciales que se deben omitir al leer los datos del archivo.

skip_comments

[in] Línea de símbolos para designar las líneas como comentarios. Si al inicio de la línea se encuentra cualquier símbolo de *skip_comments*, esta línea se considerará un comentario y no será importada.

Valor retornado

Retorna el número de líneas importadas, o -1, en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INVALID_PARAMETER (4003) - el nombre del recuadro no ha sido establecido (línea vacía o NULL);
- ERR_DATABASE_INTERNAL (5120) - error interno en la base de datos;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos.

Observación

© 2000-2025, MetaQuotes Ltd.

Si el recuadro con el nombre *table* no existe, será creado de forma automática. El nombre y el tipo de los campos en el recuadro creado serán reconocidos automáticamente usando como base los datos contenidos en el archivo.

ENUM_DATABASE_IMPORT_FLAGS

Identificador	Descripción
DATABASE_IMPORT_HEADER	La primera línea contiene los nombres de los campos del recuadro
DATABASE_IMPORT_CRLF	Para el traslado de líneas, se considera CRLF (por defecto, LF)
DATABASE_IMPORT_APPEND	Añadir datos al final de un recuadro existente
DATABASE_IMPORT_QUOTED_STRINGS	Los valores de tipo string van entre comillas dobles
DATABASE_IMPORT_COMMON_FOLDER	El archivo está ubicado en la carpeta general de todos los terminales de cliente \Terminal\Common\File.

Ejemplo de lectura de un recuadro desde un archivo creado usando el código del ejemplo en DatabaseExport :

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
  string csv_filename;
//--- obtenemos los nombres de los archivos de texto para realizar la carga desde la c
  string filenames[];
  if(FileSelectDialog("Seleccione el archivo CSV para cargar el recuadro", NULL,
                      "Text files (*.csv)|*.csv",
                      FSD_WRITE_FILE|FSD_COMMON_FOLDER, filenames, "data.csv")>0)
   {
    //--- mostramos el nombre de cada archivo seleccionado
    if(ArraySize(filenames)==1)
      csv_filename=filenames[0];
    else
     {
      Print("Error desconocido al seleccionar el archivo. Código de error ", GetLastEr
      return;
     }
   }
  else
   {
    Print("Archivo CSV no seleccionado");
    return;
   }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- creamos o abrimos la base de datos
  string db_filename="test.sqlite";
  int db=DatabaseOpen(db_filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE);
//--- comprobamos la presencia del recuadro TEST
  if(DatabaseTableExists(db, "TEST"))
    {
     //--- eliminamos el recuadro TEST
     if(!DatabaseExecute(db, "DROP TABLE IF EXISTS TEST"))
      {
       Print("Failed to drop the TEST table with code ", GetLastError());
       DatabaseClose(db);
       return;
      }
    }
  //--- importamos la entrada desde el archivo al recuadro TEST
  long imported=DatabaseImport(db, "TEST", csv_filename, DATABASE_IMPORT_HEADER|DATABA
  if(imported>0)
    {
     Print(imported," lines imported in table TEST");
     DatabasePrint(db,"SELECT * FROM TEST",DATABASE_PRINT_NO_INDEX);
    }
  else
    {
     Print("DatabaseImport() failed. Error ",GetLastError());
    }
//--- cerramos el archivo con la base de datos y comunicamos este hecho
  DatabaseClose(db);
  PrintFormat("Database: %s closed", db_filename);
 }
```

Ver también

DatabaseOpen, DatabasePrint

© 2000-2025, MetaQuotes Ltd.

DatabaseExport

Exporta un recuadro o el resultado de la ejecución de una solicitud SQL a un archivo CSV. El archivo se crea en la codificación UTF-8.

```
long  DatabaseExport(
   int           database,          // manejador de base de datos recibido en Databas
   const string  table_or_sql,      // nombre del recuadro o solicitud SQL
   const string  filename,          // nombre del archivo CSV para exportar los datos
   uint          flags,             // combinación de banderas
   const string  separator          // separador de datos en el archivo CSV
   );
```

Parámetros

database

[in] Manejador de base de datos recibido en DatabaseOpen().

table_or_sql

[in] Nombre del recuadro o solicitud SQL cuyos resultados serán exportados al archivo indicado.

filename

[in] Nombre del archivo para exportar los datos. La ruta se establece con respecto a la carpeta MQL5\Files.

flags

[in] Combinación de banderas de la enumeración ENUM_DATABASE_EXPORT_FLAGS.

separator

[in] Separador de datos. Si se indica NULL, se usará como separador el símbolo de tabulación '\t'. La línea vacía "" se considera un separador permitido, pero el archivo CSV obtenido no puede ser leído como recuadro, se tratará de un conjunto de líneas.

Valor retornado

Retorna el número de entradas exportadas o un valor negativo en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INTERNAL_ERROR (4001) - error crítico del sistema de ejecución;
- ERR_INVALID_PARAMETER (4003) - la ruta al archivo de la base de datos contiene una línea vacía o se ha establecido una combinación de banderas incompatible;
- ERR_NOT_ENOUGH_MEMORY (4004) - memoria insuficiente;
- ERR_FUNCTION_NOT_ALLOWED(4014) - el pipe indicado no está permitido;
- ERR_PROGRAM_STOPPED(4022) - la operación ha sido cancelada (el programa MQL se ha detenido);
- ERR_WRONG_FILENAME (5002) - nombre de archivo incorrecto;
- ERR_TOO_LONG_FILENAME (5003) - la ruta absoluta al archivo ha superado la longitud máxima;
- ERR_CANNOT_OPEN_FILE(5004) - error de apertura del archivo para el registro;
- ERR_FILE_WRITEERROR(5026) - error de escritura en el archivo;
- ERR_DATABASE_INTERNAL (5120) - error interno en la base de datos;

© 2000-2025, MetaQuotes Ltd.

- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_QUERY_PREPARE(5125) - error al crear la solicitud;
- ERR_DATABASE_QUERY_NOT_READONLY - se permite solo la solicitud de lectura.

Observación

Si se exportan los resultados de una solicitud, la solicitud SQL deberá comenzar con "SELECT" o "select". En otras palabras, una solicitud SQL no puede modificar la base de datos, en caso contrario, DatabaseExport() se finalizará con error.

Los valores de línea en la base de datos pueden contener el símbolo de conversión ('\r' o '\r\n'), así como el símbolo de separación de valores establecido en el parámetro *separator*. En este caso, deberemos usar obligatoriamente la bandera DATABASE_EXPORT_QUOTED_STRINGS en el parámetro flags. Si se da esta bandera, todas las líneas mostradas finalizarán en comillas dobles, y si en una línea se contienen comillas dobles, serán sustituidas por dos comillas dobles.

ENUM_DATABASE_EXPORT_FLAGS

Identificador	Descripción
DATABASE_EXPORT_HEADER	Mostrar los nombres de los campos en la primera línea
DATABASE_EXPORT_INDEX	Mostrar los números de las líneas
DATABASE_EXPORT_NO_BOM	No incorporar la etiqueta BOM al inicio del archivo (por defecto, BOM se incorpora)
DATABASE_EXPORT_CRLF	Para trasladar una línea, se utiliza CRLF (por defecto, LF)
DATABASE_EXPORT_APPEND	Añadir datos al final de un archivo existente (por defecto, el archivo se reescribe). Si el archivo no existe, será creado.
DATABASE_EXPORT_QUOTED_STRINGS	Mostrar los valores de línea entre comillas dobles.
DATABASE_EXPORT_COMMON_FOLDER	El archivo CSV se creará en la carpeta general de todos los terminales de cliente \Terminal\Common\File.

Ejemplo:

```
input int InpRates=100;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
  MqlRates rates[];
//--- registrando la hora de inicio antes de recibir las barras
```

© 2000-2025, MetaQuotes Ltd.

```
  ulong start=GetMicrosecondCount();
//--- solicitando las últimas 100 barras en el marco temporal H1
  if(CopyRates(Symbol(), PERIOD_H1, 1, InpRates, rates)<InpRates)
    {
     Print("CopyRates() failed,, Error ", GetLastError());
     return;
    }
  else
    {
     //--- cuántas barras y en qué tiempo las hemos obtenido
     PrintFormat("%s: CopyRates received %d bars in %d ms ",
                 _Symbol, ArraySize(rates), (GetMicrosecondCount()-start)/1000);
    }
//--- comparando el nombre del archivo para guardar la base de datos
  string filename=_Symbol+"_"+EnumToString(PERIOD_H1)+"_"+TimeToString(TimeCurrent())+
  StringReplace(filename, ":", "-"); // el símbolo ":" está prohibido en los nombres d
//--- abrimos/creamos la base de datos en la carpeta general de los terminales
  int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATAB
  if(db==INVALID_HANDLE)
    {
     Print("Database: ", filename, " open failed with code ", GetLastError());
     return;
    }
  else
     Print("Database: ", filename, " opened successfully");

//--- comprobamos la presencia del recuadro RATES
  if(DatabaseTableExists(db, "RATES"))
    {
     //--- eliminamos el recuadro RATES
     if(!DatabaseExecute(db, "DROP TABLE IF EXISTS RATES"))
       {
        Print("Failed to drop the RATES table with code ", GetLastError());
        DatabaseClose(db);
        return;
       }
    }
//--- creamos el recuadro RATES
  if(!DatabaseExecute(db, "CREATE TABLE RATES("
                      "SYMBOL             CHAR(10),"
                      "TIME               INT NOT NULL,"
                      "OPEN               REAL,"
                      "HIGH               REAL,"
                      "LOW                REAL,"
                      "CLOSE              REAL,"
                      "TICK_VOLUME        INT,"
                      "SPREAD             INT,"
                      "REAL_VOLUME        INT);"))
    {
```

© 2000-2025, MetaQuotes Ltd.

```
    Print("DB: ", filename, " create table RATES with code ", GetLastError());
    DatabaseClose(db);
    return;
   }
//--- mostrando la lista de todos los campos en el recuadro RATES
  if(DatabasePrint(db, "PRAGMA TABLE_INFO(RATES)", 0)<0)
   {
    PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(RATES)\") failed, error code=%d at
    DatabaseClose(db);
    return;
   }
//--- creando una solicitud parametrizada de adición de barras al recuadro RATES
  string sql="INSERT INTO RATES (SYMBOL,TIME,OPEN,HIGH,LOW,CLOSE,TICK_VOLUME,SPREAD,RE
             " VALUES (?1,?2,?3,?4,?5,?6,?7,?8,?9)"; // parámetros de la solicitud
  int request=DatabasePrepare(db, sql);
  if(request==INVALID_HANDLE)
   {
    PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
    Print("SQL request: ", sql);
    DatabaseClose(db);
    return;
   }
//--- estableciendo el valor del primer parámetro de la solicitud
  DatabaseBind(request, 0, _Symbol);
//--- rellenando la hora de comienzo antes de añadir las barras al recuadro RATES
  start=GetMicrosecondCount();
  DatabaseTransactionBegin(db);
  int total=ArraySize(rates);
  bool request_error=false;
  for(int i=0; i<total; i++)
   {
    //--- estableciendo los valores del resto de parámetros antes de añadir la entrada
    ResetLastError();
    if(!DatabaseBind(request, 1, rates[i].time))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    //--- si la anterior llamada de DatabaseBind() ha tenido éxito, establecemos el si
    if(!request_error && !DatabaseBind(request, 2, rates[i].open))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 3, rates[i].high))
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 4, rates[i].low))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 5, rates[i].close))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 6, rates[i].tick_volume))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 7, rates[i].spread))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 8, rates[i].real_volume))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }

    //--- ejecutando la solicitud de inserción de una entrada y comprobando si es erró
    if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_MO
     {
      PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
      DatabaseFinalize(request);
      request_error=true;
      break;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
     //--- reseteando la solicitud a su estado inicial antes de la siguiente actualizac
     if(!request_error && !DatabaseBind(request))
      {
       PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
    } //--- ya hemos finalizado, se ha pasado por todas las barras

//--- ¿cómo han salido las transacciones?
  if(request_error)
    {
     PrintFormat("Table RATES: failed to add %d bars ", ArraySize(rates));
     DatabaseTransactionRollback(db);
     DatabaseClose(db);
     return;
    }
  else
    {
     DatabaseTransactionCommit(db);
     PrintFormat("Table RATES: added %d bars in %d ms",
                 ArraySize(rates), (GetMicrosecondCount()-start)/1000);
    }
//--- guardando el recuadro RATES en un archivo CSV
  string csv_filename=Symbol()+".csv";
  long saved=DatabaseExport(db, "SELECT * FROM RATES", csv_filename, DATABASE_EXPORT_F
  if(saved>0)
    Print("Table RATES saved in ", Symbol(), ".csv");
  else
    Print("DatabaseExport() failed. Error ", GetLastError());
//--- cerramos el archivo con la base de datos y comunicamos este hecho
  DatabaseClose(db);
  PrintFormat("Database: %s created and closed", filename);
```

Ver también

DatabasePrint, DatabaseImport

© 2000-2025, MetaQuotes Ltd.

DatabasePrint

Imprime el recuadro o resultado de la ejecución de una solicitud SQL en el diario de expertos.

```
long  DatabasePrint(
   int           database,          // manejador de base de datos recibido en Database
   const string  table_or_sql,      // recuadro o solicitud SQL
   uint          flags              // combinación de banderas
   );
```

Parámetros

database

[in] Manejador de base de datos recibido en DatabaseOpen().

table_or_sql

[in] Nombre del recuadro o texto de la solicitud SQL cuyos resultados se muestran en el diario de expertos.

flags

[in] Combinación de banderas que determina el formato de la muestra. Las banderas se determinan de la forma siguiente:

DATABASE_PRINT_NO_HEADER - no mostrar los nombres de las columnas del recuadro (nombres de los campos)
DATABASE_PRINT_NO_INDEX - no mostrar los números de las líneas
DATABASE_PRINT_NO_FRAME - no mostrar el marco que separa el encabezado y los datos
DATABASE_PRINT_STRINGS_RIGHT - alinear las líneas a la derecha.

Si flags=0, se mostrarán las columnas y las líneas; los encabezados y los datos se separarán con un marco, y las líneas se alinearán a la izquierda.

Valor retornado

Retorna el número de líneas exportadas, o -1, en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INTERNAL_ERROR (4001) - error crítico del sistema de ejecución;
- ERR_NOT_ENOUGH_MEMORY (4004) - memoria insuficiente;
- ERR_DATABASE_INTERNAL (5120) - error interno en la base de datos;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;

Observación

Si en el diario se muestran los resultados de una solicitud, la solicitud SQL deberá comenzar con "SELECT" o "select". En otras palabras, una solicitud SQL no puede modificar la base de datos, en caso contrario, DatabasePrint() se finalizará con error.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   string filename="departments.sqlite";
//--- Creamos o abrimos la base de datos en la carpeta general del terminal
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATAB
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }

//--- creamos el recuadro COMPANY
   if(!CreataTableCompany(db))
     {
      DatabaseClose(db);
      return;
     }
//--- creamos el recuadro DEPARTMENT
   if(!CreataTableDepartment(db))
     {
      DatabaseClose(db);
      return;
     }

//--- mostramos la lista con todos los campos en el recuadro COMPANY y DEPARTMENT
   PrintFormat("Try to print request \"PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DE
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)", 0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO()\") failed, error code=%d", GetL
      DatabaseClose(db);
      return;
     }
//--- mostramos en el log el recuadro COMPANY
   PrintFormat("Try to print request \"SELECT * from COMPANY\"");
   if(DatabasePrint(db, "SELECT * from COMPANY", 0)<0)
     {
      Print("DatabasePrint failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- texto de la solicitud para combinar los recuadros COMPANY y DEPARTMENT
   string request="SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT "
                  "ON COMPANY.ID = DEPARTMENT.EMP_ID";
//--- mostramos el resultado de la combinación de los recuadros
   PrintFormat("Try to print request \"SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUT
   if(DatabasePrint(db, request, 0)<0)
     {
      Print("DatabasePrint failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- cerramos la base de datos
   DatabaseClose(db);
  }
/*
Conclusión:
Try to print request "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)"
#| cid name    type     notnull dflt_value pk
-+-----------------------------------------
1|   0 ID      INT            1             1
2|   1 NAME    TEXT           1             0
3|   2 AGE     INT            1             0
4|   3 ADDRESS CHAR(50)       0             0
5|   4 SALARY  REAL           0             0
#| cid name   type     notnull dflt_value pk
-+-----------------------------------------
1|   0 ID     INT            1             1
2|   1 DEPT   CHAR(50)       1             0
3|   2 EMP_ID INT            1             0
Try to print request "SELECT * from COMPANY"
#| ID NAME  AGE ADDRESS     SALARY
-+-------------------------------
1|  1 Paul   32 California 25000.0
2|  2 Allen  25 Texas      15000.0
3|  3 Teddy  23 Norway     20000.0
4|  4 Mark   25 Rich-Mond  65000.0
5|  5 David  27 Texas      85000.0
6|  6 Kim    22 South-Hall 45000.0
7|  7 James  24 Houston    10000.0
Try to print request "SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMEN
#| EMP_ID NAME  DEPT
-+-------------------------
1|      1 Paul  IT Billing
2|      2 Allen Engineering
3|        Teddy
4|        Mark
5|        David
6|        Kim
7|      7 James Finance
*/
//+------------------------------------------------------------------+
//| Crea el recuadro COMPANY                                         |
//+------------------------------------------------------------------+
bool CreataTableCompany(int database)
  {
//--- si el recuadro COMPANY existe, lo eliminamos
   if(DatabaseTableExists(database, "COMPANY"))
     {
      //--- eliminamos el recuadro
      if(!DatabaseExecute(database, "DROP TABLE COMPANY"))
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          Print("Failed to drop table COMPANY with code ", GetLastError());
          return(false);
         }
      }
//--- eliminamos el recuadro COMPANY
   if(!DatabaseExecute(database, "CREATE TABLE COMPANY("
                         "ID INT PRIMARY KEY     NOT NULL,"
                         "NAME           TEXT    NOT NULL,"
                         "AGE            INT     NOT NULL,"
                         "ADDRESS        CHAR(50),"
                         "SALARY         REAL );"))
      {
       Print("DB: create table COMPANY failed with code ", GetLastError());
       return(false);
      }

//--- insertamos los datos en el recuadro COMPANY
   if(!DatabaseExecute(database, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VAI
                         "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'A
                         "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'T
                         "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'M
                         "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (5, 'D
                         "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (6, 'P
                         "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7, 'J
      {
       Print("COMPANY insert failed with code ", GetLastError());
       return(false);
      }
//--- éxito
   return(true);
  }
//+------------------------------------------------------------------+
//| Crea el recuadro DEPARTMENT                                      |
//+------------------------------------------------------------------+
bool CreataTableDepartment(int database)
  {
//--- si el recuadro DEPARTMENT existe, lo eliminamos
   if(DatabaseTableExists(database, "DEPARTMENT"))
     {
      //--- eliminamos el recuadro
      if(!DatabaseExecute(database, "DROP TABLE DEPARTMENT"))
        {
         Print("Failed to drop table DEPARTMENT  with code ", GetLastError());
         return(false);
        }
     }
//--- creamos el recuadro DEPARTMENT
   if(!DatabaseExecute(database, "CREATE TABLE DEPARTMENT ("
```

© 2000-2025, MetaQuotes Ltd.

```
                        "ID      INT PRIMARY KEY   NOT NULL,"
                        "DEPT    CHAR(50)          NOT NULL,"
                        "EMP_ID  INT               NOT NULL);"))
     {
      Print("DB: create table DEPARTMENT failed with code ", GetLastError());
      return(false);
     }

//--- insertamos los datos en el recuadro DEPARTMENT
   if(!DatabaseExecute(database, "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (1, '
                        "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (2, 'Engineerir
                        "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (3, 'Finance',
     {
      Print("DEPARTMENT insert failed with code ", GetLastError());
      return(false);
     }
//--- éxito
   return(true);
  }
//+------------------------------------------------------------------+
```

Ver también

DatabaseExport, DatabaseImport

© 2000-2025, MetaQuotes Ltd.

DatabaseTableExists

Comprueba la presencia de un recuadro en la base de datos.

```
bool  DatabaseTableExists(
   int     database,      // manejador de base de datos recibido en DatabaseOpen
   string  table          // nombre del recuadro
   );
```

Parámetros

database

[in] Manejador de base de datos recibido en DatabaseOpen().

table

[in] Nombre del recuadro.

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INVALID_PARAMETER (4003) - el nombre del recuadro no ha sido establecido (línea vacía o NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - error de conversión de la solicitud en una línea UTF-8;
- ERR_DATABASE_INTERNAL (5120) - error interno en la base de datos;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_EXECUTE (5124) - error de ejecución de la solicitud;
- ERR_DATABASE_NO_MORE_DATA (5126) - el recuadro no existe (no es un error, finalización normal).

Ver también

DatabasePrepare, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabaseExecute

Ejecuta una solicitud a la base de datos indicada.

```
bool  DatabaseExecute(
   int     database,      // manejador de base de datos recibido en DatabaseOpen
   string  sql,           // solicitud SQL
   );
```

Parámetros

database

[in] Manejador de base de datos recibido en DatabaseOpen().

sql

[in] Solicitud SQL.

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INTERNAL_ERROR (4001) - error crítico del sistema de ejecución;
- ERR_INVALID_PARAMETER (4003) - el parámetro sql contiene una línea vacía;
- ERR_NOT_ENOUGH_MEMORY (4004) - memoria insuficiente;
- ERR_WRONG_STRING_PARAMETER (5040) - error de conversión de la solicitud en una línea UTF-8;
- ERR_DATABASE_INTERNAL (5120) - error interno en la base de datos;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_EXECUTE (5124) - error de ejecución de la solicitud.

Example:

```
//--- symbol statistics
struct Symbol_Stats
  {
   string            name;             // symbol name
   int               trades;           // number of trades for the symbol
   double            gross_profit;     // total profit for the symbol
   double            gross_loss;       // total loss for the symbol
   double            total_commission; // total commission for the symbol
   double            total_swap;       // total swaps for the symbol
   double            total_profit;     // total profit excluding swaps and commissions
   double            net_profit;       // net profit taking into account swaps and cor
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swap
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
```

© 2000-2025, MetaQuotes Ltd.

```
  };

//--- Magic Number statistics
struct Magic_Stats
  {
   long              magic;            // EA's Magic Number
   int               trades;           // number of trades for the symbol
   double            gross_profit;     // total profit for the symbol
   double            gross_loss;       // total loss for the symbol
   double            total_commission; // total commission for the symbol
   double            total_swap;       // total swaps for the symbol
   double            total_profit;     // total profit excluding swaps and commissions
   double            net_profit;       // net profit taking into account swaps and cor
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swap
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };

//--- entry hour statistics
struct Hour_Stats
  {
   char              hour_in;          // market entry hour
   int               trades;           // number of trades in this entry hour
   double            volume;           // volume of trades in this entry hour
   double            gross_profit;     // total profit in this entry hour
   double            gross_loss;       // total loss in this entry hour
   double            net_profit;       // net profit taking into account swaps and cor
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swap
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };

int ExtDealsTotal=0;;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
```

© 2000-2025, MetaQuotes Ltd.

```
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_stats.sqlite";
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
   PrintFormat("Deals in the trading history: %d ", ExtDealsTotal);

//--- get trading statistics per symbols
   int request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then (r.gross_profit+r.gro
                               "   (case when r.trades != 0 then r.win_trades*100.0/r.
                               "   (case when r.trades != 0 then r.loss_trades*100.0/r
                               "   r.gross_profit/r.win_trades as average_profit,"
                               "   r.gross_loss/r.loss_trades as average_loss,"
                               "   (case when r.gross_loss!=0.0 then r.gross_profit/(-
                               "FROM "
                               "   ("
                               "   SELECT SYMBOL,"
                               "   sum(case when entry =1 then 1 else 0 end) as trades
                               "   sum(case when profit > 0 then profit else 0 end) as
                               "   sum(case when profit < 0 then profit else 0 end) as
                               "   sum(swap) as total_swap,"
                               "   sum(commission) as total_commission,"
                               "   sum(profit) as total_profit,"
                               "   sum(profit+swap+commission) as net_profit,"
                               "   sum(case when profit > 0 then 1 else 0 end) as win_
                               "   sum(case when profit < 0 then 1 else 0 end) as loss
                               "   FROM DEALS "
                               "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                               "   GROUP BY SYMBOL"
                               "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Symbol_Stats stats[], symbol_stats;
   ArrayResize(stats, ExtDealsTotal);
   int i=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- get records from request results
   for(; DatabaseReadBind(request, symbol_stats) ; i++)
     {
      stats[i].name=symbol_stats.name;
      stats[i].trades=symbol_stats.trades;
      stats[i].gross_profit=symbol_stats.gross_profit;
      stats[i].gross_loss=symbol_stats.gross_loss;
      stats[i].total_commission=symbol_stats.total_commission;
      stats[i].total_swap=symbol_stats.total_swap;
      stats[i].total_profit=symbol_stats.total_profit;
      stats[i].net_profit=symbol_stats.net_profit;
      stats[i].win_trades=symbol_stats.win_trades;
      stats[i].loss_trades=symbol_stats.loss_trades;
      stats[i].expected_payoff=symbol_stats.expected_payoff;
      stats[i].win_percent=symbol_stats.win_percent;
      stats[i].loss_percent=symbol_stats.loss_percent;
      stats[i].average_profit=symbol_stats.average_profit;
      stats[i].average_loss=symbol_stats.average_loss;
      stats[i].profit_factor=symbol_stats.profit_factor;
     }
   ArrayResize(stats, i);
   Print("Trade statistics by Symbol");
   ArrayPrint(stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- get trading statistics for Expert Advisors by Magic Numbers
   request=DatabasePrepare(db, "SELECT r.*,"
                           "   (case when r.trades != 0 then (r.gross_profit+r.gross_l
                           "   (case when r.trades != 0 then r.win_trades*100.0/r.trad
                           "   (case when r.trades != 0 then r.loss_trades*100.0/r.tra
                           "   r.gross_profit/r.win_trades as average_profit,"
                           "   r.gross_loss/r.loss_trades as average_loss,"
                           "   (case when r.gross_loss!=0.0 then r.gross_profit/(-r.gr
                           "FROM "
                           "   ("
                           "   SELECT MAGIC,"
                           "   sum(case when entry =1 then 1 else 0 end) as trades,"
                           "   sum(case when profit > 0 then profit else 0 end) as gro
                           "   sum(case when profit < 0 then profit else 0 end) as gro
                           "   sum(swap) as total_swap,"
                           "   sum(commission) as total_commission,"
                           "   sum(profit) as total_profit,"
                           "   sum(profit+swap+commission) as net_profit,"
                           "   sum(case when profit > 0 then 1 else 0 end) as win_trad
                           "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                           "   FROM DEALS "
                           "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
```

© 2000-2025, MetaQuotes Ltd.

```
                              "   GROUP BY MAGIC"
                              "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Magic_Stats EA_stats[], magic_stats;
   ArrayResize(EA_stats, ExtDealsTotal);
   i=0;
//--- print
   for(; DatabaseReadBind(request, magic_stats) ; i++)
     {
      EA_stats[i].magic=magic_stats.magic;
      EA_stats[i].trades=magic_stats.trades;
      EA_stats[i].gross_profit=magic_stats.gross_profit;
      EA_stats[i].gross_loss=magic_stats.gross_loss;
      EA_stats[i].total_commission=magic_stats.total_commission;
      EA_stats[i].total_swap=magic_stats.total_swap;
      EA_stats[i].total_profit=magic_stats.total_profit;
      EA_stats[i].net_profit=magic_stats.net_profit;
      EA_stats[i].win_trades=magic_stats.win_trades;
      EA_stats[i].loss_trades=magic_stats.loss_trades;
      EA_stats[i].expected_payoff=magic_stats.expected_payoff;
      EA_stats[i].win_percent=magic_stats.win_percent;
      EA_stats[i].loss_percent=magic_stats.loss_percent;
      EA_stats[i].average_profit=magic_stats.average_profit;
      EA_stats[i].average_loss=magic_stats.average_loss;
      EA_stats[i].profit_factor=magic_stats.profit_factor;
     }
   ArrayResize(EA_stats, i);
   Print("Trade statistics by Magic Number");
   ArrayPrint(EA_stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- make sure that hedging system for open position management is used on the accou
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
     {
      //--- deals cannot be transformed to trades using a simple method through transa
      DatabaseClose(db);
      return;
     }

//--- now create the TRADES table based on the DEALS table
   if(!CreateTableTrades(db))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      DatabaseClose(db);
      return;
     }
//--- fill in the TRADES table using an SQL query based on DEALS table data
   if(DatabaseTableExists(db, "DEALS"))
      //--- populate the TRADES table
      if(!DatabaseExecute(db, "INSERT INTO TRADES(TIME_IN,HOUR_IN,TICKET,TYPE,VOLUME,S
                              "SELECT "
                              "   d1.time as time_in,"
                              "   d1.hour as hour_in,"
                              "   d1.position_id as ticket,"
                              "   d1.type as type,"
                              "   d1.volume as volume,"
                              "   d1.symbol as symbol,"
                              "   d1.price as price_in,"
                              "   d2.time as time_out,"
                              "   d2.price as price_out,"
                              "   d1.commission+d2.commission as commission,"
                              "   d2.swap as swap,"
                              "   d2.profit as profit "
                              "FROM DEALS d1 "
                              "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                              "WHERE d1.entry=0 AND d2.entry=1     "))
        {
         Print("DB: fillng the table TRADES failed with code ", GetLastError());
         return;
        }

//--- get trading statistics by market entry hours
   request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then (r.gross_profit+r.gross_l
                               "   (case when r.trades != 0 then r.win_trades*100.0/r.trad
                               "   (case when r.trades != 0 then r.loss_trades*100.0/r.tra
                               "   r.gross_profit/r.win_trades as average_profit,"
                               "   r.gross_loss/r.loss_trades as average_loss,"
                               "   (case when r.gross_loss!=0.0 then r.gross_profit/(-r.gr
                              "FROM "
                               "   ("
                               "   SELECT HOUR_IN,"
                               "   count() as trades,"
                               "   sum(volume) as volume,"
                               "   sum(case when profit > 0 then profit else 0 end) as gro
                               "   sum(case when profit < 0 then profit else 0 end) as gro
                               "   sum(profit) as net_profit,"
                               "   sum(case when profit > 0 then 1 else 0 end) as win_trad
                               "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                               "   FROM TRADES "
                               "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                               "   GROUP BY HOUR_IN"
```

© 2000-2025, MetaQuotes Ltd.

```
                              "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Hour_Stats hours_stats[], h_stats;
   ArrayResize(hours_stats, ExtDealsTotal);
   i=0;
//--- print
   for(; DatabaseReadBind(request, h_stats) ; i++)
     {
      hours_stats[i].hour_in=h_stats.hour_in;
      hours_stats[i].trades=h_stats.trades;
      hours_stats[i].volume=h_stats.volume;
      hours_stats[i].gross_profit=h_stats.gross_profit;
      hours_stats[i].gross_loss=h_stats.gross_loss;
      hours_stats[i].net_profit=h_stats.net_profit;
      hours_stats[i].win_trades=h_stats.win_trades;
      hours_stats[i].loss_trades=h_stats.loss_trades;
      hours_stats[i].expected_payoff=h_stats.expected_payoff;
      hours_stats[i].win_percent=h_stats.win_percent;
      hours_stats[i].loss_percent=h_stats.loss_percent;
      hours_stats[i].average_profit=h_stats.average_profit;
      hours_stats[i].average_loss=h_stats.average_loss;
      hours_stats[i].profit_factor=h_stats.profit_factor;
     }
   ArrayResize(hours_stats, i);
   Print("Trade statistics by entry hour");
   ArrayPrint(hours_stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- close database
   DatabaseClose(db);
   return;
  }
/*
Deals in the trading history: 2771
Trade statistics by Symbol
      [name] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap] [tot
[0] "AUDUSD"      112      503.20000   -568.00000           -8.83000    -24.64000
[1] "EURCHF"      125      607.71000   -956.85000          -11.77000    -45.02000
[2] "EURJPY"      127     1078.49000  -1057.83000          -10.61000    -45.76000
[3] "EURUSD"      233     1685.60000  -1386.80000          -41.00000    -83.76000
[4] "GBPCHF"      125     1881.37000  -1424.72000          -22.60000    -51.56000
[5] "GBPJPY"      127     1943.43000  -1776.67000          -18.84000    -52.46000
```

© 2000-2025, MetaQuotes Ltd.

```
[6] "GBPUSD"      121     1668.50000  -1438.20000           -7.96000    -49.93000
[7] "USDCAD"       99      405.28000   -475.47000           -8.68000    -31.68000
[8] "USDCHF"      206     1588.32000  -1241.83000          -17.98000    -65.92000
[9] "USDJPY"      107      464.73000   -730.64000          -35.12000    -34.24000

Trade statistics by Magic Number
    [magic] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap] [tota
[0]     100      242     2584.80000  -2110.00000          -33.36000    -93.53000
[1]     200      254     3021.92000  -2834.50000          -29.45000    -98.22000
[2]     300      250     2489.08000  -2381.57000          -34.37000    -96.58000
[3]     400      224     1272.50000  -1283.00000          -24.43000    -64.80000
[4]     500      198     1141.23000  -1051.91000          -27.66000    -63.36000
[5]     600      214     1317.10000  -1396.03000          -34.12000    -68.48000

Trade statistics by entry hour
     [hour_in] [trades] [volume] [gross_profit] [gross_loss] [net_profit] [win_trades]
[ 0]         0       50  5.00000      336.51000   -747.47000   -410.96000           21
[ 1]         1       20  2.00000      102.56000    -57.20000     45.36000           12
[ 2]         2        6  0.60000       38.55000    -14.60000     23.95000            5
[ 3]         3       38  3.80000      173.84000   -200.15000    -26.31000           22
[ 4]         4       60  6.00000      361.44000   -389.40000    -27.96000           27
[ 5]         5       32  3.20000      157.43000   -179.89000    -22.46000           20
[ 6]         6       18  1.80000       95.59000   -162.33000    -66.74000           11
[ 7]         7       14  1.40000       38.48000   -134.30000    -95.82000            9
[ 8]         8       42  4.20000      368.48000   -322.30000     46.18000           24
[ 9]         9      118 11.80000     1121.62000   -875.21000    246.41000           72
[10]        10      206 20.60000     2280.59000  -2021.80000    258.79000          115
[11]        11      138 13.80000     1377.02000   -994.18000    382.84000           84
[12]        12      152 15.20000     1247.56000  -1463.80000   -216.24000           84
[13]        13       64  6.40000      778.27000   -516.22000    262.05000           36
[14]        14       62  6.20000      536.93000   -427.47000    109.46000           38
[15]        15       50  5.00000      699.92000   -413.00000    286.92000           28
[16]        16       88  8.80000      778.55000   -514.00000    264.55000           51
[17]        17       76  7.60000      533.92000  -1019.46000   -485.54000           44
[18]        18       52  5.20000      237.17000   -246.78000     -9.61000           24
[19]        19       52  5.20000      407.67000   -150.36000    257.31000           30
[20]        20       18  1.80000       65.92000    -89.09000    -23.17000            9
[21]        21       10  1.00000       41.86000    -32.38000      9.48000            7
[22]        22       14  1.40000       45.55000    -83.72000    -38.17000            6
[23]        23        2  0.20000        1.20000     -1.90000     -0.70000            1
*/

//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(database, "DEALS"))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       return(false);
      }
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "HOUR        INT,"
                          "REASON      INT);"))
        {
         Print("DB: create the DEALS table  failed with code ", GetLastError());
         return(false);
        }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   ExtDealsTotal=HistoryDealsTotal();
//--- add deals to the table
   if(!InsertDeals(database))
      return(false);
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop the DEALS table with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was exec
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number (Expert Advisor ID)
   long     reason;              // deal execution reason or source
   char     hour;                // deal execution hour
   MqlDateTime time_strusture;
//--- go through all deals and add them to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- lock the database before executing transactions
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      TimeToStruct(time, time_strusture);
      hour= (char)time_strusture.hour;
      //--- add each deal to the table using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
```

© 2000-2025, MetaQuotes Ltd.

```
                                        "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                        deal_ticket, order_ticket, position_ticket, tir
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%d with code %d", __FUNCTION__, deal_
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- roll back all transactions and unlock the database
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastErro
      return(false);
     }
//--- all transactions have been performed successfully - record changes and unlock th
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the TRADES table                                         |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- if the TRADES table already exists, delete it
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- check if the table exists
   if(!DatabaseTableExists(database, "TRADES"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "HOUR_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
         Print("DB: create the TRADES table failed with code ", GetLastError());
         return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
         }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
```

Ver también

DatabasePrepare, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabasePrepare

Crea el manejador de una solicitud, que después puede ser ejecutada con la ayuda de DatabaseRead().

```
int  DatabasePrepare(
   int     database,      // manejador de base de datos recibido en DatabaseOpen
   string  sql,           // solicitud SQL
           ...            // parámetros de la solicitud
   );
```

Parámetros

database

[in] Manejador de base de datos recibido en DatabaseOpen().

sql

[in] Solicitud SQL que puede contener los parámetros autosustituibles con los nombres ?1,?2,...

...

[in] Parámetros de solicitud sustituidos automáticamente.

Valor retornado

Si la ejecución tiene éxito, retorna el manejador de la solicitud SQL, en el caso contrario, retorna INVALID_HANDLE. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INVALID_PARAMETER (4003) - la ruta al archivo de la base de datos contiene una línea vacía o se ha establecido una combinación de banderas incompatible;
- ERR_NOT_ENOUGH_MEMORY (4004) - memoria insuficiente;
- ERR_WRONG_STRING_PARAMETER (5040) - error de conversión de la solicitud en una línea UTF-8;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - se ha superado el número máximo permitido de objetos Database;
- ERR_DATABASE_PREPARE (5125) - error al crear la solicitud.

Observación

La función DatabasePrepare() no ejecuta la solicitud a la base de datos. Su cometido es comprobar si los parámetros de la solicitud son correctos y retornar - según los resultados de la comprobación - el manejador para la ejecución de la solicitud SQL. La propia solicitud se realizará con la primera llamada de DatabaseRead().

Example:

```
//--- Structure to store the deal
struct Deal
  {
   ulong             ticket;           // DEAL_TICKET
   long              order_ticket;     // DEAL_ORDER
   long              position_ticket;  // DEAL_POSITION_ID
   datetime          time;             // DEAL_TIME
   char              type;             // DEAL_TYPE
```

© 2000-2025, MetaQuotes Ltd.

```
   char              entry;            // DEAL_ENTRY
   string            symbol;           // DEAL_SYMBOL
   double            volume;           // DEAL_VOLUME
   double            price;            // DEAL_PRICE
   double            profit;           // DEAL_PROFIT
   double            swap;             // DEAL_SWAP
   double            commission;       // DEAL_COMMISSION
   long              magic;            // DEAL_MAGIC
   char              reason;           // DEAL_REASON
  };
//--- Structure to store the trade: the order of members corresponds to the position i
struct Trade
  {
   datetime          time_in;          // entry time
   ulong             ticket;           // position ID
   char              type;             // buy or sell
   double            volume;           // volume
   string            symbol;           // symbol
   double            price_in;         // entry price
   datetime          time_out;         // exit time
   double            price_out;        // exit price
   double            commission;       // entry and exit commission
   double            swap;             // swap
   double            profit;           // profit or loss
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_trades.sqlite"
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
```

© 2000-2025, MetaQuotes Ltd.

```
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("Deals in the trading history: %d ", deals_total);
//--- add deals to the table
   if(!InsertDeals(db))
      return;
//--- show the first 10 deals
   Deal deals[], deal;
   ArrayResize(deals, 10);
   int request=DatabasePrepare(db, "SELECT * FROM DEALS");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   int i;
   for(i=0; DatabaseReadBind(request, deal); i++)
     {
      if(i>=10)
         break;
      deals[i].ticket=deal.ticket;
      deals[i].order_ticket=deal.order_ticket;
      deals[i].position_ticket=deal.position_ticket;
      deals[i].time=deal.time;
      deals[i].type=deal.type;
      deals[i].entry=deal.entry;
      deals[i].symbol=deal.symbol;
      deals[i].volume=deal.volume;
      deals[i].price=deal.price;
      deals[i].profit=deal.profit;
      deals[i].swap=deal.swap;
      deals[i].commission=deal.commission;
      deals[i].magic=deal.magic;
      deals[i].reason=deal.reason;
     }
//--- print the deals
   if(i>0)
     {
      ArrayResize(deals, i);
      PrintFormat("The first %d deals:", i);
      ArrayPrint(deals);
     }

//--- delete request after use
   DatabaseFinalize(request);

//--- make sure that hedging system for open position management is used on the accou
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- deals cannot be transformed to trades using a simple method through transa
       DatabaseClose(db);
       return;
      }

//--- now create the TRADES table based on the DEALS table
   if(!CreateTableTrades(db))
      {
       DatabaseClose(db);
       return;
      }
//--- fill in the TRADES table using an SQL query based on DEALS table data
   ulong start=GetMicrosecondCount();
   if(DatabaseTableExists(db, "DEALS"))
      //--- populate the TRADES table
      if(!DatabaseExecute(db, "INSERT INTO TRADES(TIME_IN,TICKET,TYPE,VOLUME,SYMBOL,PR
                          "SELECT "
                          "   d1.time as time_in,"
                          "   d1.position_id as ticket,"
                          "   d1.type as type,"
                          "   d1.volume as volume,"
                          "   d1.symbol as symbol,"
                          "   d1.price as price_in,"
                          "   d2.time as time_out,"
                          "   d2.price as price_out,"
                          "   d1.commission+d2.commission as commission,"
                          "   d2.swap as swap,"
                          "   d2.profit as profit "
                          "FROM DEALS d1 "
                          "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                          "WHERE d1.entry=0 AND d2.entry=1     "))
        {
         Print("DB: fillng the TRADES table failed with code ", GetLastError());
         return;
        }
   ulong transaction_time=GetMicrosecondCount()-start;

//--- show the first 10 deals
   Trade trades[], trade;
   ArrayResize(trades, 10);
   request=DatabasePrepare(db, "SELECT * FROM TRADES");
   if(request==INVALID_HANDLE)
      {
       Print("DB: ", filename, " request failed with code ", GetLastError());
       DatabaseClose(db);
       return;
      }
   for(i=0; DatabaseReadBind(request, trade); i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       if(i>=10)
          break;
       trades[i].time_in=trade.time_in;
       trades[i].ticket=trade.ticket;
       trades[i].type=trade.type;
       trades[i].volume=trade.volume;
       trades[i].symbol=trade.symbol;
       trades[i].price_in=trade.price_in;
       trades[i].time_out=trade.time_out;
       trades[i].price_out=trade.price_out;
       trades[i].commission=trade.commission;
       trades[i].swap=trade.swap;
       trades[i].profit=trade.profit;
      }
//--- print trades
   if(i>0)
     {
      ArrayResize(trades, i);
      PrintFormat("\r\nThe first %d trades:", i);
      ArrayPrint(trades);
      PrintFormat("Filling the TRADES table took %.2f milliseconds",double(transactior
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- close the database
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2741
   The first 10 deals:
       [ticket] [order_ticket] [position_ticket]              [time] [type] [entry] [s
   [0] 34429573              0                 0 2019.09.05 22:39:59      2        0 "'
   [1] 34432127       51447238          51447238 2019.09.06 06:00:03      0        0 "U
   [2] 34432128       51447239          51447239 2019.09.06 06:00:03      1        0 "U
   [3] 34432450       51447565          51447565 2019.09.06 07:00:00      0        0 "E
   [4] 34432456       51447571          51447571 2019.09.06 07:00:00      1        0 "A
   [5] 34432879       51448053          51448053 2019.09.06 08:00:00      1        0 "U
   [6] 34432888       51448064          51448064 2019.09.06 08:00:00      0        0 "U
   [7] 34435147       51450470          51450470 2019.09.06 10:30:00      1        0 "E
   [8] 34435152       51450476          51450476 2019.09.06 10:30:00      0        0 "C
   [9] 34435154       51450479          51450479 2019.09.06 10:30:00      1        0 "E

   The first 10 trades:
                [time_in] [ticket] [type] [volume] [symbol] [price_in]           [time
   [0] 2019.09.06 06:00:03 51447238      0  0.10000 "USDCAD"    1.32320 2019.09.06 18:
   [1] 2019.09.06 06:00:03 51447239      1  0.10000 "USDCHF"    0.98697 2019.09.06 18:
```

© 2000-2025, MetaQuotes Ltd.

```
   [2] 2019.09.06 07:00:00 51447565      0  0.10000 "EURUSD"    1.10348 2019.09.09 03:
   [3] 2019.09.06 07:00:00 51447571      1  0.10000 "AUDUSD"    0.68203 2019.09.09 03:
   [4] 2019.09.06 08:00:00 51448053      1  0.10000 "USDCHF"    0.98701 2019.09.06 18:
   [5] 2019.09.06 08:00:00 51448064      0  0.10000 "USDJPY"  106.96200 2019.09.06 18:
   [6] 2019.09.06 10:30:00 51450470      1  0.10000 "EURUSD"    1.10399 2019.09.06 14:
   [7] 2019.09.06 10:30:00 51450476      0  0.10000 "GBPUSD"    1.23038 2019.09.06 14:
   [8] 2019.09.06 10:30:00 51450479      1  0.10000 "EURJPY"  118.12000 2019.09.06 14:
   [9] 2019.09.06 10:30:00 51450480      0  0.10000 "GBPJPY"  131.65300 2019.09.06 14:
   Filling the TRADES table took 12.51 milliseconds
*/
//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(database, "DEALS"))
     {
      return(false);
     }
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
        {
         Print("DB: create the DEALS table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop the DEALS table  with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was exec
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number (Expert Advisor ID)
   long     reason;              // deal execution reason or source
//--- go through all deals and add them to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- lock the database before executing transactions
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
```

© 2000-2025, MetaQuotes Ltd.

```
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- add each deal to the table using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                       deal_ticket, order_ticket, position_ticket, tim
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%d with code %d", __FUNCTION__, deal_
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- roll back all transactions and unlock the database
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code %d", __FUNCTION__, GetLastEr
      return(false);
     }
//--- all transactions have been performed successfully - record changes and unlock th
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the TRADES table                                         |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- if the TRADES table already exists, delete it
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- check if the table exists
   if(!DatabaseTableExists(database, "TRADES"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         Print("DB: create the TRADES table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
```

Ver también

DatabaseExecute, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabaseReset

Resetea la solicitud a su estado inicial, igual que tras la llamada de DatabasePrepare().

```
int  DatabaseReset(
   int  request      // manejador de solicitud recibido en DatabasePrepare
   );
```

Parámetros

request

[in] Manejador de la solicitud obtenido en DatabasePrepare().

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código de error, use GetLastError(), posibles respuestas:

- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- códigos de error de SQLite, comenzando por ERR_DATABASE_ERROR(5601).

Observación

La función DatabaseReset() ha sido pensada para ejecutar varias veces una solicitud con diferentes valores en los parámetros. Por ejemplo, al añadir datos a un recuadro de forma masiva con ayuda del comando INSERT, para cada entrada se deberá formar el conjunto propio correspondiente de valores de cada campo.

A diferencia de DatabasePrepare(), la llamada de DatabaseReset() no provoca la compilación de la línea con los comandos SQL en una nueva solicitud, por eso, DatabaseReset() se ejecuta bastante más rápido que DatabasePrepare().

DatabaseReset() se usa conjuntamente con las funciones DatabaseBind() y/o DatabaseBindArray(), si es necesario cambiar los valores de los parámetros de una solicitud después de la ejecución de DatabaseRead(). Esto significa que, antes de establecer los nuevos valores de los parámetros de la solicitud (antes del bloque de llamadas DatabaseBind/DatabaseBindArray), es necesario llamar DatabaseReset() para resetearla al estado inicial. La propia solicitud parametrizada debe ser creada con la ayuda de DatabasePrepare().

DatabaseReset(), al igual que DatabasePrepare(), no realiza la solicitud a la base de datos. La propia ejecución de la solicitud tiene lugar al llamar a DatabaseRead() o DatabaseReadBind().

La llamada de DatabaseReset() no provoca el reseteo de los valores de los parámetros en la solicitud si han sido establecidos mediante la llamada de DatabaseBind()/DatabaseBindArray(), es decir, los parámetros mantienen sus valores. De esta forma, podremos modificar el valor de un solo parámetro, no tendremos necesidad de escribir de nuevo todos los parámtros de la solicitud después de llamar a DatabaseReset().

No podemos transmitir a DatabaseReset() un puntero de solicitud que ha sido eliminado antes de esto con la ayuda de DatabaseFinalize(). Dicha acción provocaría un error.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- creamos o abrimos la base de datos
   string filename="symbols.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");
//--- si el recuadro SYMBOLS existe, lo eliminamos
   if(DatabaseTableExists(db, "SYMBOLS"))
     {
      //--- eliminamos el recuadro
      if(!DatabaseExecute(db, "DROP TABLE SYMBOLS"))
        {
         Print("Failed to drop table SYMBOLS with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- creando el recuadro SYMBOLS
   if(!DatabaseExecute(db, "CREATE TABLE SYMBOLS("
                       "NAME           TEXT    NOT NULL,"
                       "DESCRIPTION    TEXT            ,"
                       "PATH           TEXT            ,"
                       "SPREAD         INT             ,"
                       "POINT          REAL    NOT NULL,"
                       "DIGITS         INT     NOT NULL,"
                       "JSON           BLOB );"))
     {
      Print("DB: ", filename, " create table failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- mostrando la lista de todos los campos en el recuadro SYMBOLS
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(SYMBOLS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(SYMBOLS)\") failed, error code=%d
      DatabaseClose(db);
      return;
     }

//--- creando una solicitud parametrizada de adición de símbolos al recuadro SYMBOLS
   string sql="INSERT INTO SYMBOLS (NAME,DESCRIPTION,PATH,SPREAD,POINT,DIGITS,JSON)"
              " VALUES (?1,?2,?3,?4,?5,?6,?7);"; // parámetros de la solicitud
   int request=DatabasePrepare(db, sql);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }

//--- iteramos por todos los símbolos y los añadimos al recuadro SYMBOLS
   int symbols=SymbolsTotal(false);
   bool request_error=false;
   DatabaseTransactionBegin(db);
   for(int i=0; i<symbols; i++)
     {
      //--- estableciendo los valores de los parámetros antes de añadir el símbolo
      ResetLastError();
      string symbol=SymbolName(i, false);
      if(!DatabaseBind(request, 0, symbol))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      //--- si la anterior llamada de DatabaseBind() ha tenido éxito, establecemos el
      if(!DatabaseBind(request, 1, SymbolInfoString(symbol, SYMBOL_DESCRIPTION)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 2, SymbolInfoString(symbol, SYMBOL_PATH)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 3, SymbolInfoInteger(symbol, SYMBOL_SPREAD)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 4, SymbolInfoDouble(symbol, SYMBOL_POINT)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 5, SymbolInfoInteger(symbol, SYMBOL_DIGITS)))
```

© 2000-2025, MetaQuotes Ltd.

```
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 6, GetSymBolAsJson(symbol)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }

      //--- ejecutando la solicitud de inserción de una entrada y comprobando si es er
      if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
        {
         PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
      else
         PrintFormat("%d: added %s", i+1, symbol);
      //--- reseteando la solicitud a su estado inicial antes de la siguiente actualiz
      if(!DatabaseReset(request))
        {
         PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
     } //--- ya hemos finalizado, se ha pasado por todos los ticks

//--- ¿cómo han salido las transacciones?
   if(request_error)
     {
      PrintFormat("Table SYMBOLS: failed to add %d symbols", symbols);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table SYMBOLS: added %d symbols",symbols);
     }

//--- guardando el recuadro SYMBOLS en un archivo CSV
   string csv_filename="symbols.csv";
   if(DatabaseExport(db, "SELECT * FROM SYMBOLS", csv_filename,
```

© 2000-2025, MetaQuotes Ltd.

```
                     DATABASE_EXPORT_HEADER|DATABASE_EXPORT_INDEX|DATABASE_EXPORT_QUOT
      Print("Database: table SYMBOLS saved in ", csv_filename);
   else
      Print("Database: DatabaseExport(\"SELECT * FROM SYMBOLS\") failed with code", Ge

//--- cerramos el archivo con la base de datos y comunicamos este hecho
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
//+------------------------------------------------------------------+
//| Retorna las especificaciones del símbolo como JSON               |
//+------------------------------------------------------------------+
string GetSymBolAsJson(string symbol)
  {
//--- sangrías
   string indent1=Indent(1);
   string indent2=Indent(2);
   string indent3=Indent(3);
//---
   int digits=(int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   string json="{"+
               "\n"+indent1+"\"ConfigSymbols\":["+
               "\n"+indent2+"{"+
               "\n"+indent3+"\"Symbol\":\""+symbol+"\","+
               "\n"+indent3+"\"Path\":\""+SymbolInfoString(symbol, SYMBOL_PATH)+"\","+
               "\n"+indent3+"\"CurrencyBase\":\""+SymbolInfoString(symbol, SYMBOL_CURR
               "\n"+indent3+"\"CurrencyProfit\":\""+SymbolInfoString(symbol, SYMBOL_CU
               "\n"+indent3+"\"CurrencyMargin\":\""+SymbolInfoString(symbol, SYMBOL_CU
               "\n"+indent3+"\"ColorBackground\":\""+ColorToString((color)SymbolInfoIn
               "\n"+indent3+"\"Digits\":\""+IntegerToString(SymbolInfoInteger(symbol,
               "\n"+indent3+"\"Point\":\""+DoubleToString(SymbolInfoDouble(symbol, SYM
               "\n"+indent3+"\"TickBookDepth\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"ChartMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"TradeMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"TradeCalcMode\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"OrderMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"CalculationMode\":\""+IntegerToString(SymbolInfoIntege
               "\n"+indent3+"\"ExecutionMode\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"ExpirationMode\":\""+IntegerToString(SymbolInfoInteger
               "\n"+indent3+"\"FillFlags\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"ExpirFlags\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"Spread\":\""+IntegerToString(SymbolInfoInteger(symbol,
               "\n"+indent3+"\"TickValue\":\""+StringFormat("%G", (SymbolInfoDouble(sy
               "\n"+indent3+"\"TickSize\":\""+StringFormat("%G", (SymbolInfoDouble(syr
               "\n"+indent3+"\"ContractSize\":\""+StringFormat("%G",(SymbolInfoDouble
               "\n"+indent3+"\"StopsLevel\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"VolumeMin\":\""+StringFormat("%G",(SymbolInfoDouble(syr
               "\n"+indent3+"\"VolumeMax\":\""+StringFormat("%G",(SymbolInfoDouble(syr
               "\n"+indent3+"\"VolumeStep\":\""+StringFormat("%G",(SymbolInfoDouble(sy
```

© 2000-2025, MetaQuotes Ltd.

```
                "\n"+indent3+"\"VolumeLimit\":\""+StringFormat("%G",(SymbolInfoDouble(s
                "\n"+indent3+"\"SwapMode\":\""+IntegerToString(SymbolInfoInteger(symbol
                "\n"+indent3+"\"SwapLong\":\""+StringFormat("%G",(SymbolInfoDouble(symb
                "\n"+indent3+"\"SwapShort\":\""+StringFormat("%G",(SymbolInfoDouble(syr
                "\n"+indent3+"\"Swap3Day\":\""+IntegerToString(SymbolInfoInteger(symbol
                "\n"+indent2+"}"+
                "\n"+indent1+"]"+
                "\n}";
   return(json);
  }
//+------------------------------------------------------------------+
//| Forma la sangría a partir de espacios                            |
//+------------------------------------------------------------------+
string Indent(const int number, const int characters=3)
  {
   int length=number*characters;
   string indent=NULL;
   StringInit(indent, length, ' ');
   return indent;
  }
/*
 Resultado:
  Database: symbols.sqlite opened successfully
  #| cid name        type notnull dflt_value pk
  -+-----------------------------------------
  1|   0 NAME        TEXT       1             0
  2|   1 DESCRIPTION TEXT       0             0
  3|   2 PATH        TEXT       0             0
  4|   3 SPREAD      INT        0             0
  5|   4 POINT       REAL       1             0
  6|   5 DIGITS      INT        1             0
  7|   6 JSON        BLOB       0             0
  1: added EURUSD
  2: added GBPUSD
  3: added USDCHF
  ...
  82: added USDCOP
  83: added USDARS
  84: added USDCLP
  Table SYMBOLS: added 84 symbols
  Database: table SYMBOLS saved in symbols.csv
  Database: symbols.sqlite created and closed
*/
```

Ver también

DatabasePrepare, DatabaseBind, DatabaseBindArray, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabaseBind

Establece el valor de un parámetro en la solicitud.

```
bool  DatabaseBind(
   int   request,      // manejador de la solicitud creada en DatabasePrepare
   int   index,        // índice del parámetro en la solicitud
   T     value         // valor del parámetro de tipo simple
   );
```

Parámetros

request

[in] Manejador de la solicitud creada en DatabasePrepare().

index

[in] Índice del parámetro en la solicitud para la que se debe establecer el valor. La numeración parte desde el cero.

value

[in] Valor del parámetro que se debe establecer. Tipos permitidos: bool, char, uchar, short, ushart, int, uint, color, datetime, long, ulong, float, double, string.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INVALID_PARAMETER (4003) - tipo no soportado;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_NOT_READY (5128) - no es posible usar la función para la solicitud en estos momentos. La solicitud se está ejecutando o ya se ha realizado, se debe llamar a DatabaseReset().

Observación

La función se debe usar solo cuando una solicitud SQL contiene los valores parametrizables "?" o "?N", donde N indica el número del parámetro (comenzando por la unidad). En este caso, además, la indexación de los parámetros en DatabaseBind() comienza a partir de cero.

Por ejmplo:

```
INSERT INTO table VALUES (?,?,?)
SELECT * FROM table WHERE id=?
```

La función se puede llamar justo después de que la solicitud parametrizada haya sido creada en DatabasePrepare(), o bien tras resetear la solicitud a su estado inicial con la ayuda de DatabaseReset().

Use esta función junto con DatabaseReset() para ejecutar la solicitud el número necesario de veces con diferentes valores de parámetros.

La función ha sido pensada para trabajar con los parámetros de tipos simples. Si tiene que comparar un parámetro con una matriz, use la función DatabaseBindArray().

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlTick ticks[];
//--- registrando la hora de inicio antes de recibir los ticks
   uint start=GetTickCount();
//--- solicitando la historia de ticks de un día
   ulong to=TimeCurrent()*1000;
   ulong from=to-PeriodSeconds(PERIOD_D1)*1000;
   if(CopyTicksRange(_Symbol, ticks, COPY_TICKS_ALL, from, to)==-1)
     {
      PrintFormat("%s: CopyTicksRange(%s - %s) failed, error=%d",
                  _Symbol, TimeToString(datetime(from/1000)), TimeToString(datetime(to
      return;
     }
   else
     {
      //--- cuántos ticks y en qué tiempo los hemos obtenido
      PrintFormat("%s: CopyTicksRange received %d ticks in %d ms (from %s to %s)",
                  _Symbol, ArraySize(ticks), GetTickCount()-start,
                  TimeToString(datetime(from/1000)), TimeToString(datetime(to/1000)));
     }

//--- comparando el nombre del archivo para guardar la base de datos
   string filename=_Symbol+" "+TimeToString(datetime(from/1000))+" - "+TimeToString(da
   StringReplace(filename, ":", "."); // el símbolo ":" está prohibido en los nombres
//--- abrimos/creamos la base de datos en la carpeta general de los terminales
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("Database: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");

//--- creando el recuadro TICKS
   if(!DatabaseExecute(db, "CREATE TABLE TICKS("
                       "SYMBOL             CHAR(10),"
                       "TIME               INT NOT NULL,"
                       "BID                REAL,"
                       "ASK                REAL,"
                       "LAST               REAL,"
                       "VOLUME             INT,"
                       "TIME_MSC           INT,"
```

© 2000-2025, MetaQuotes Ltd.

```
                        "VOLUME_REAL        REAL);"))
     {
      Print("DB: ", filename, " create table TICKS failed with code ", GetLastError())
      DatabaseClose(db);
      return;
     }
//--- mostrando la lista de todos los campos en el recuadro TICKS
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(TICKS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(TICKS)\") failed, error code=%d a
      DatabaseClose(db);
      return;
     }
//--- creando una solicitud parametrizada de adición de ticks al recuadro TICKS
   string sql="INSERT INTO TICKS (SYMBOL,TIME,BID,ASK,LAST,VOLUME,TIME_MSC,VOLUME_REAL
              " VALUES (?1,?2,?3,?4,?5,?6,?7,?8)"; // parámetros de la solicitud
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }
//--- estableciendo el valor del primer parámetro de la solicitud
   DatabaseBind(request, 0, _Symbol);
//--- rellenando la hora de comienzo antes de añadir los ticks al recuadro TICKS
   start=GetTickCount();
   DatabaseTransactionBegin(db);
   int total=ArraySize(ticks);
   bool request_error=false;
   for(int i=0; i<total; i++)
     {
      //--- estableciendo los valores del resto de parámetros antes de añadir la entra
      ResetLastError();
      if(!DatabaseBind(request, 1, ticks[i].time))
        {
         PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
         PrintFormat("Tick #%d line=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
      //--- si la anterior llamada de DatabaseBind() ha tenido éxito, establecemos el
      if(!request_error && !DatabaseBind(request, 2, ticks[i].bid))
        {
         PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
         PrintFormat("Tick #%d line=%d", i+1, __LINE__);
         request_error=true;
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   if(!request_error && !DatabaseBind(request, 3, ticks[i].ask))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
   if(!request_error && !DatabaseBind(request, 4, ticks[i].last))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
   if(!request_error && !DatabaseBind(request, 5, ticks[i].volume))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
   if(!request_error && !DatabaseBind(request, 6, ticks[i].time_msc))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
   if(!request_error && !DatabaseBind(request, 7, ticks[i].volume_real))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }

   //--- ejecutando la solicitud de inserción de una entrada y comprobando si es er
   if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_
      {
       PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
   //--- reseteando la solicitud a su estado inicial antes de la siguiente actualiz
   if(!request_error && !DatabaseReset(request))
      {
       PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
     } //--- ya hemos finalizado, se ha pasado por todos los ticks

//--- ¿cómo han salido las transacciones?
   if(request_error)
     {
      PrintFormat("Table TICKS: failed to add %d ticks ", ArraySize(ticks));
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table TICKS: added %d ticks in %d ms",
                  ArraySize(ticks), GetTickCount()-start);
     }

//--- cerramos el archivo con la base de datos y comunicamos este hecho
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
/*
 Resultado:
  EURUSD: CopyTicksRange received 268061 ticks in 47 ms (from 2020.03.18 12:40 to 2020
  Database: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite opened successfully
  #| cid name        type     notnull dflt_value pk
  -+-----------------------------------------------
  1|   0 SYMBOL      CHAR(10)       0              0
  2|   1 TIME        INT            1              0
  3|   2 BID         REAL           0              0
  4|   3 ASK         REAL           0              0
  5|   4 LAST        REAL           0              0
  6|   5 VOLUME      INT            0              0
  7|   6 TIME_MSC    INT            0              0
  8|   7 VOLUME_REAL REAL           0              0
  Table TICKS: added 268061 ticks in 797 ms
  Database: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite created and closed
  OnCalculateCorrelation=0.87 2020.03.19 13:00:  EURUSD vs GBPUSD  PERIOD_M30
*/
```

Ver también

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBindArray

© 2000-2025, MetaQuotes Ltd.

DatabaseBindArray

Establece una matriz como valor del parámetro.

```
bool  DatabaseBind(
   int   request,       // manejador de la solicitud creada en DatabasePrepare
   int   index,         // índice del parámetro en la solicitud
   T&    array[]        // valor del parámetro en forma de matriz
   );
```

Parámetros

request

[in] Manejador de la solicitud creada en DatabasePrepare().

index

[in] Índice del parámetro en la solicitud para la que se debe establecer el valor. La numeración parte desde el cero.

array[]

[in] Matriz que se debe establecer como valor del parámetro de la solicitud.

Valor retornado

Retorna true en el caso de éxito, de lo contrario, false. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INVALID_PARAMETER (4003) - tipo no soportado;
- ERR_ARRAY_BAD_SIZE (4011) - el tamaño de la matriz en bytes supera el valor INT_MAX;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_NOT_READY (5128) - no es posible usar la función para la solicitud en estos momentos (La solicitud ya se está ejecutando o ha sido finalizada, es necesario llamar a DatabaseReset).

Observación

La función se debe usar solo cuando una solicitud SQL contiene los valores parametrizables "?" o "?N", donde N indica el número del parámetro (comenzando por la unidad). En este caso, además, la indexación de los parámetros en DatabaseBindArray() comienza a partir de cero.

Por ejmplo:

```
INSERT INTO table VALUES (?,?,?)
```

La función se puede llamar justo después de que la solicitud parametrizada haya sido creada en DatabasePrepare(), o bien tras resetear la solicitud a su estado inicial con la ayuda de DatabaseReset().

Use esta función junto con DatabaseReset() para ejecutar la solicitud el número necesario de veces con diferentes valores de parámetros.

Ejemplo:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- abriendo ventana de diálogo para seleccionar los archivos con la extensión DAT
   string selected_files[];
   if(!FileSelectDialog("Seleccione los archivos para la carga", NULL,
                        "Data files (*.dat)|*.dat|All files (*.*)|*.*",
                        FSD_ALLOW_MULTISELECT, selected_files, "tester.dat")>0)
     {
      Print("Files not selected. Exit");
      return;
     }
//--- obteniendo el tamaño de los archivos
   ulong filesize[];
   int filehandle[];
   int files=ArraySize(selected_files);
   ArrayResize(filesize, files);
   ZeroMemory(filesize);
   ArrayResize(filehandle, files);
   double total_size=0;
   for(int i=0; i<files; i++)
     {
      filehandle[i]=FileOpen(selected_files[i], FILE_READ|FILE_BIN);
      if(filehandle[i]!=INVALID_HANDLE)
        {
         filesize[i]=FileSize(filehandle[i]);
         //PrintFormat("%d, %s handle=%d %d bytes", i, selected_files[i], filehandle[i
         total_size+=(double)filesize[i];
        }
     }
//--- comprobando el tamaño total de los archivos
   if(total_size==0)
     {
      PrintFormat("Total files size is 0. Exit");
      return;
     }

//--- Creamos o abrimos la base de datos en la carpeta general del terminal
   string filename="dat_files.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");
//--- si el recuadro FILES existe, lo eliminamos
```

© 2000-2025, MetaQuotes Ltd.

```
   if(DatabaseTableExists(db, "FILES"))
     {
      //--- eliminamos el recuadro
      if(!DatabaseExecute(db, "DROP TABLE FILES"))
        {
         Print("Failed to drop table FILES with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- creamos el recuadro FILES
   if(!DatabaseExecute(db, "CREATE TABLE FILES("
                       "NAME           TEXT NOT NULL,"
                       "SIZE           INT  NOT NULL,"
                       "PERCENT_SIZE   REAL NOT NULL,"
                       "DATA           BLOB NOT NULL);"))
     {
      Print("DB: failed to create table FILES with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- mostrando la lista de todos los campos en el recuadro FILES
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(FILES)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(FILES)\") failed, error code=%d a
      DatabaseClose(db);
      return;
     }

//--- creando una solicitud parametrizada de adición de archivos al recuadro FILES
   string sql="INSERT INTO FILES (NAME,SIZE,PERCENT_SIZE,DATA)"
              " VALUES (?1,?2,?3,?4);"; // parámetros de la solicitud
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }

//--- iteramos por todos los archivos y los añadimos al recuadro FILES
   bool request_error=false;
   DatabaseTransactionBegin(db);
   int count=0;
   uint size;
   for(int i=0; i<files; i++)
     {
      if(filehandle[i]!=INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       char data[];
       size=FileReadArray(filehandle[i], data);
       if(size==0)
         {
          PrintFormat("FileReadArray(%s) failed with code %d", selected_files[i], Ge
          continue;
         }

       count++;
       //--- estableciendo los valores de los parámetros antes de añadir un archivo
       if(!DatabaseBind(request, 0, selected_files[i]))
         {
          PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
          request_error=true;
          break;
         }
       if(!DatabaseBind(request, 1, size))
         {
          PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
          request_error=true;
          break;
         }
       if(!DatabaseBind(request, 2, double(size)*100./total_size))
         {
          PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
          request_error=true;
          break;
         }
       if(!DatabaseBindArray(request, 3, data))
         {
          PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
          request_error=true;
          break;
         }
       //--- ejecutando la solicitud de inserción de una entrada y comprobando si es
       if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
         {
          PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
          DatabaseFinalize(request);
          request_error=true;
          break;
         }
       else
          PrintFormat("%d. %s: %d bytes", count, selected_files[i],size);
       //--- reseteando la solicitud a su estado inicial antes de la siguiente actua
       if(!DatabaseReset(request))
         {
          PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
            DatabaseFinalize(request);
            request_error=true;
            break;
           }
        }
     }
//--- ¿cómo han salido las transacciones?
   if(request_error)
     {
      PrintFormat("Table FILES: failed to add %d files", count);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table FILES: added %d files", count);
     }

//--- cerramos el archivo con la base de datos y comunicamos este hecho
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
```

Ver también

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBind

© 2000-2025, MetaQuotes Ltd.

DatabaseRead

Ejecuta el paso a la siguiente entrada en el resultado de la solicitud.

```
bool  DatabaseRead(
   int  request      // manejador de solicitud recibido en DatabasePrepare
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INVALID_PARAMETER (4003) - el nombre del recuadro no ha sido establecido (línea vacía o NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - error de conversión de la solicitud en una línea UTF-8;
- ERR_DATABASE_INTERNAL (5120) - error interno en la base de datos;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_EXECUTE (5124) - error de ejecución de la solicitud;
- ERR_DATABASE_NO_MORE_DATA (5126) - el recuadro no existe (no es un error, finalización normal).

Ver también

DatabasePrepare, DatabaseReadBind

© 2000-2025, MetaQuotes Ltd.

DatabaseReadBind

Pasa a la siguiente entrada y lee los datos en la estructura de esta.

```
bool  DatabaseReadBind(
   int    request,           // manejador de la solicitud creada en DatabasePrepare
   void&  struct_object      // enlace a la estructura para leer la entrada
   );
```

Parámetros

request

[in] Manejador de la solicitud creada en DatabasePrepare().

struct_object

[out] Enlace a la estructura en la que se leerán los datos de la entrada actual. La estructura deberá tener como miembros solo tipos numéricos y/o string (las matrices no están permitidas), y no puede ser heredera.

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INVALID_PARAMETER (4003) - el nombre del recuadro no ha sido establecido (línea vacía o NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - error de conversión de la solicitud en una línea UTF-8;
- ERR_DATABASE_INTERNAL (5120) - error interno en la base de datos;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_EXECUTE (5124) - error de ejecución de la solicitud;
- ERR_DATABASE_NO_MORE_DATA (5126) - el recuadro no existe (no es un error, finalización normal).

Observación

El número de campos en la estructura *struct_object* no puede superar DatabaseColumnsCount(). Si el número de campos en la estructura *struct_object* es inferior al número de campos en la entrada, se realizará solo la lectura parcial. El resto de datos se podrá obtener explícitamente con la ayuda de las funciones correspondientes DatabaseColumnText(), DatabaseColumnInteger(), etcétera.

Ejemplo:

```
struct Person
  {
   int               id;
   string            name;
   int               age;
   string            address;
   double            salary;
  };
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int db;
   string filename="company.sqlite";
//--- open
   db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATABASE_
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- if the table COMPANY exists then drop the table
   if(DatabaseTableExists(db, "COMPANY"))
     {
      //--- delete the table
      if(!DatabaseExecute(db, "DROP TABLE COMPANY"))
        {
         Print("Failed to drop table COMPANY with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- create table
   if(!DatabaseExecute(db, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
     {
      Print("DB: ", filename, " create table failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- insert data
   if(!DatabaseExecute(db, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'A
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'T
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'M
     {
      Print("DB: ", filename, " insert failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- prepare the request
```

© 2000-2025, MetaQuotes Ltd.

```
   int request=DatabasePrepare(db, "SELECT * FROM COMPANY WHERE SALARY>15000");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- print records
   Person person;
   Print("Persons with salary > 15000:");
   for(int i=0; DatabaseReadBind(request, person); i++)
      Print(i, ":  ", person.id, " ", person.name, " ", person.age, " ", person.addres
//--- delete request after use
   DatabaseFinalize(request);

   Print("Some statistics:");
//--- prepare new request about total salary
   request=DatabasePrepare(db, "SELECT SUM(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double total_salary;
      DatabaseColumnDouble(request, 0, total_salary);
      Print("Total salary=", total_salary);
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- prepare new request about average salary
   request=DatabasePrepare(db, "SELECT AVG(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      ResetLastError();
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double aver_salary;
      DatabaseColumnDouble(request, 0, aver_salary);
      Print("Average salary=", aver_salary);
     }
//--- delete request after use
```

© 2000-2025, MetaQuotes Ltd.

```
   DatabaseFinalize(request);

//--- close database
   DatabaseClose(db);
  }
//+------------------------------------------------------------------
/*
Output:
Persons with salary > 15000:
0:  1 Paul 32 California 25000.0
1:  3 Teddy 23 Norway 20000.0
2:  4 Mark 25 Rich-Mond  65000.0
Some statistics:
Total salary=125000.0
Average salary=31250.0
*/
```

Ver también

DatabasePrepare, DatabaseRead

© 2000-2025, MetaQuotes Ltd.

DatabaseFinalize

Elimina la solicitud creada en DatabasePrepare().

```
void  DatabaseFinalize(
   int  request      // manejador de solicitud recibido en DatabasePrepare
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

Valor retornado

No.

Observación

Si el manejador no es válido, la función mostrará el error ERR_DATABASE_INVALID_HANDLE. Es posible comprobar el error con la ayuda de GetLastError().

Ver también

DatabasePrepare, DatabaseExecute

© 2000-2025, MetaQuotes Ltd.

DatabaseTransactionBegin

Comienza la ejecución de una transacción.

```
bool  DatabaseTransactionBegin(
   int  database      // manejador de base de datos recibido en DatabaseOpen
   );
```

Parámetros

database

[in] Manejador de base de datos recibido en DatabaseOpen().

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INTERNAL_ERROR (4001) - error crítico del sistema de ejecución;
- ERR_INVALID_PARAMETER (4003) - el parámetro sql contiene una línea vacía;
- ERR_NOT_ENOUGH_MEMORY (4004) - memoria insuficiente;
- ERR_WRONG_STRING_PARAMETER (5040) - error de conversión de la solicitud en una línea UTF-8;
- ERR_DATABASE_INTERNAL (5120) - error interno en la base de datos;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_EXECUTE (5124) - error de ejecución de la solicitud.

Observación

La función DatabaseTransactionBegin() se debe llamar antes de ejecutar la transacción. Cualquier transacción debe comenzar con la llamada de DatabaseTransactionBegin() y terminar con la llamada de DatabaseTransactionCommit().

Example:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=AccountInfoString(ACCOUNT_SERVER) +"_"+IntegerToString(AccountInfo
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("Deals in the trading history: %d ", deals_total);

//--- measure the transaction execution speed using DatabaseTransactionBegin/DatabaseT
   ulong start=GetMicrosecondCount();
   bool fast_transactions=true;
   InsertDeals(db, fast_transactions);
   double fast_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit

//--- delete the DEALS table, and then create it again
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
//--- create a new DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }

//--- test again, this time without using DatabaseTransactionBegin/DatabaseTransactior
   fast_transactions=false;
   start=GetMicrosecondCount();
   InsertDeals(db, fast_transactions);
   double slow_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit
//--- report gain in time
   PrintFormat("Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided acc
//--- close the database
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2737
```

© 2000-2025, MetaQuotes Ltd.

```
   Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit: time=48.5 r
   Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit: time=25818.
   Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided acceleration by
*/
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop table DEALS with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
        {
         Print("DB: create table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool InsertDeals(int database, bool begintransaction=true)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was exec
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number
   long     reason;              // deal execution reason or source
//--- go through all deals and add to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
//--- if fast transaction performance method is used
   if(begintransaction)
     {
      // --- lock the database before executing transactions
      DatabaseTransactionBegin(database);
     }
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- add each deal using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                       deal_ticket, order_ticket, position_ticket, tim
      if(!DatabaseExecute(database, request_text))
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("%s: failed to insert deal #%dwith code %d", __FUNCTION__, deal_t
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- if fast transaction performance method is used
      if(begintransaction)
        {
         //--- roll back all transactions and unlock the database
         DatabaseTransactionRollback(database);
        }
      Print("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastError());
      return(false);
     }
//--- if fast transaction performance method is used
   if(begintransaction)
     {
      //--- all transactions have been performed successfully - record changes and unl
      DatabaseTransactionCommit(database);
     }
//--- successful completion
   return(true);
  }
//+------------------------------------------------------------------+
```

Ver también

DatabaseExecute, DatabasePrepare, DatabaseTransactionCommit, DatabaseTransactionRollback

© 2000-2025, MetaQuotes Ltd.

DatabaseTransactionCommit

Finaliza la ejecución de una transacción.

```
bool  DatabaseTransactionCommit(
   int  database      // manejador de base de datos recibido en DatabaseOpen
   );
```

Parámetros

database

[in] Manejador de base de datos recibido en DatabaseOpen().

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INTERNAL_ERROR (4001) - error crítico del sistema de ejecución;
- ERR_INVALID_PARAMETER (4003) - el parámetro sql contiene una línea vacía;
- ERR_NOT_ENOUGH_MEMORY (4004) - memoria insuficiente;
- ERR_WRONG_STRING_PARAMETER (5040) - error de conversión de la solicitud en una línea UTF-8;
- ERR_DATABASE_INTERNAL (5120) - error interno en la base de datos;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_EXECUTE (5124) - error de ejecución de la solicitud.

Observación

La función DatabaseTransactionCommit() finaliza todas las transacciones que han sido ejecutadas después de llamar la función DatabaseBeginTransaction(). Para que se realice con éxito, cualquier transacción debe comenzar con la llamada de DatabaseTransactionBegin() y terminar con la llamada de DatabaseTransactionCommit().

Ver también

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionRollback

© 2000-2025, MetaQuotes Ltd.

DatabaseTransactionRollback

Ejecuta el retroceso de una transacción.

```
bool  DatabaseTransactionRollback(
   int  database      // manejador de base de datos recibido en DatabaseOpen
   );
```

Parámetros

database

[in] Manejador de base de datos recibido en DatabaseOpen().

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_INTERNAL_ERROR (4001) - error crítico del sistema de ejecución;
- ERR_INVALID_PARAMETER (4003) - el parámetro sql contiene una línea vacía;
- ERR_NOT_ENOUGH_MEMORY (4004) - memoria insuficiente;
- ERR_WRONG_STRING_PARAMETER (5040) - error de conversión de la solicitud en una línea UTF-8;
- ERR_DATABASE_INTERNAL (5120) - error interno en la base de datos;
- ERR_DATABASE_INVALID_HANDLE (5121) - manejador no válido de la base de datos;
- ERR_DATABASE_EXECUTE (5124) - error de ejecución de la solicitud.

Observación

La llamada de DatabaseTransactionRollback() cancela todas las transacciones ejecutadas después de la llamada de la función DatabaseTransactionBegin(). La función DatabaseTransactionRollback() es necesaria para retroceder en los cambios de la base de datos, si han surgido errores durante la ejecución de una transacción.

Ver también

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionCommit

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnsCount

Obtiene el número de campos en una solicitud.

```
int  DatabaseColumnsCount(
   int  request      // manejador de solicitud recibido en DatabasePrepare
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

Valor retornado

Número de campos, o -1 en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_DATABASE_INVALID_HANDLE (5121) - manejador de solicitud no válido.

Observación

Para obtener el número de campos de una solicitud creada en DatabasePrepare(), no es necesario llamar la función DatabaseRead(). Para las demás funciones del tipo DatabaseColumnXXX(), es necesario llamar de forma preliminar a DatabaseRead().

Ver también

DatabasePrepare, DatabaseFinalize, DatabaseClose

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnName

Obtiene el nombre de un campo según el número.

```
bool  DatabaseColumnName(
   int       request,     // manejador de solicitud recibido en DatabasePrepare
   int       column,      // número de campo en la solicitud
   string&   name         // enlace a la variable para obtener el nombre del campo
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

column

[in] Número de campo en la solicitud. La numeración comienza desde cero y no puede superar el valor DatabaseColumnsCount() - 1.

name

[out] Variable para registrar el nombre del campo.

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_DATABASE_INVALID_HANDLE (5121) - código de manejador no válido;
- ERR_DATABASE_NO_MORE_DATA (5126) - el índice column es mayor que el valor DatabaseColumnsCount() -1.

Observación

El valor solo se puede obtener en el caso de que se haya realizado de forma preliminar para la solicitud *request* aunque sea una llamada de DatabaseRead().

Ver también

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnType

Obtiene el tipo de un campo según el número.

```
ENUM_DATABASE_FIELD_TYPE  DatabaseColumnType(
   int  request,     // manejador de solicitud recibido en DatabasePrepare
   int  column       // número de campo en la solicitud
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

column

[in] Número de campo en la solicitud. La numeración comienza desde cero y no puede superar el valor DatabaseColumnsCount() - 1.

Valor retornado

Retorna el tipo de campo desde la enumeración ENUM_DATABASE_FIELD_TYPE. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_DATABASE_INVALID_HANDLE (5121) - código de manejador no válido;
- ERR_DATABASE_NO_MORE_DATA (5126) - el índice column es mayor que el valor DatabaseColumnsCount() -1.

Observación

El valor solo se puede obtener en el caso de que se haya realizado de forma preliminar para la solicitud *request* aunque sea una llamada de DatabaseRead().

ENUM_DATABASE_FIELD_TYPE

Identificador	Descripción
DATABASE_FIELD_TYPE_INVALID	Error al obtener el tipo; es posible obtener el código de error con la ayuda de int GetLastError()
DATABASE_FIELD_TYPE_INTEGER	Tipo entero
DATABASE_FIELD_TYPE_FLOAT	Tipo real
DATABASE_FIELD_TYPE_TEXT	Tipo string
DATABASE_FIELD_TYPE_BLOB	Tipo binario
DATABASE_FIELD_TYPE_NULL	Tipo especial NULL

Ver también

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnSize

Obtiene el tamaño del campo en bytes.

```
int  DatabaseColumnSize(
   int  request,     // manejador de solicitud recibido en DatabasePrepare
   int  column       // número de campo en la solicitud
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

column

[in] Número de campo en la solicitud. La numeración comienza desde cero y no puede superar el valor DatabaseColumnsCount() - 1.

Valor retornado

En caso de éxito, retorna el tamaño del campo en bytes, de lo contrario, -1. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_DATABASE_INVALID_HANDLE (5121) - código de manejador no válido;
- ERR_DATABASE_NO_MORE_DATA (5126) - el índice column es mayor que el valor DatabaseColumnsCount() -1.

Observación

El valor solo se puede obtener en el caso de que se haya realizado de forma preliminar para la solicitud *request* aunque sea una llamada de DatabaseRead().

Ver también

DatabasePrepare, DatabaseColumnBlob, DatabaseColumnsCount, DatabaseColumnName, DatabaseColumnType

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnText

Obtiene de la entrada actual el valor del campo en forma de línea.

```
bool  DatabaseColumnText(
   int      request,     // manejador de solicitud recibido en DatabasePrepare
   int      column,      // número de campo en la solicitud
   string&  value        // enlace a la variable para obtener el valor
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

column

[in] Número de campo en la solicitud. La numeración comienza desde cero y no puede superar el valor DatabaseColumnsCount() - 1.

value

[out] Enlace a la variable para registrar el valor del campo.

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_DATABASE_INVALID_HANDLE (5121) - código de manejador no válido;
- ERR_DATABASE_NO_MORE_DATA (5126) - el índice column es mayor que el valor DatabaseColumnsCount() -1.

Observación

El valor solo se puede obtener en el caso de que se haya realizado de forma preliminar para la solicitud request aunque sea una llamada de DatabaseRead().

Para leer el valor de la siguiente entrada, hay que llamar de forma preliminar a DatabaseRead().

Ver también

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnInteger

Obtiene de la entrada actual un valor del tipo int.

```
bool  DatabaseColumnInteger(
   int    request,     // manejador de solicitud recibido en DatabasePrepare
   int    column,      // número de campo en la solicitud
   int&   value        // enlace a la variable para obtener el valor
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

column

[in] Número de campo en la solicitud. La numeración comienza desde cero y no puede superar el valor DatabaseColumnsCount() - 1.

value

[out] Enlace a la variable para registrar el valor del campo.

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_DATABASE_INVALID_HANDLE (5121) - código de manejador no válido;
- ERR_DATABASE_NO_MORE_DATA (5126) - el índice column es mayor que el valor DatabaseColumnsCount() -1.

Observación

El valor solo se puede obtener en el caso de que se haya realizado de forma preliminar para la solicitud request aunque sea una llamada de DatabaseRead().

Para leer el valor de la siguiente entrada, hay que llamar de forma preliminar a DatabaseRead().

Ver también

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnLong

Obtiene de la entrada actual un valor del tipo long.

```
bool  DatabaseColumnLong(
   int    request,     // manejador de solicitud recibido en DatabasePrepare
   int    column,      // número de campo en la solicitud
   long&  value        // enlace a la variable para obtener el valor
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

column

[in] Número de campo en la solicitud. La numeración comienza desde cero y no puede superar el valor DatabaseColumnsCount() - 1.

value

[out] Enlace a la variable para registrar el valor del campo.

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_DATABASE_INVALID_HANDLE (5121) - código de manejador no válido;
- ERR_DATABASE_NO_MORE_DATA (5126) - el índice column es mayor que el valor DatabaseColumnsCount() -1.

Observación

El valor solo se puede obtener en el caso de que se haya realizado de forma preliminar para la solicitud request aunque sea una llamada de DatabaseRead().

Para leer el valor de la siguiente entrada, hay que llamar de forma preliminar a DatabaseRead().

Ver también

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnDouble

Obtiene de la entrada actual un valor del tipo double.

```
bool  DatabaseColumnDouble(
   int      request,     // manejador de solicitud recibido en DatabasePrepare
   int      column,      // número de campo en la solicitud
   double&  value        // enlace a la variable para obtener el valor
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

column

[in] Número de campo en la solicitud. La numeración comienza desde cero y no puede superar el valor DatabaseColumnsCount() - 1.

value

[out] Enlace a la variable para registrar el valor del campo.

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_DATABASE_INVALID_HANDLE (5121) - código de manejador no válido;
- ERR_DATABASE_NO_MORE_DATA (5126) - el índice column es mayor que el valor DatabaseColumnsCount() -1.

Observación

El valor solo se puede obtener en el caso de que se haya realizado de forma preliminar para la solicitud request aunque sea una llamada de DatabaseRead().

Para leer el valor de la siguiente entrada, hay que llamar de forma preliminar a DatabaseRead().

Ver también

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnBlob

Obtiene de la entrada actual el valor del campo en forma de matriz.

```
bool  DatabaseColumnBlob(
   int    request,     // manejador de solicitud recibido en DatabasePrepare
   int    column,      // número de campo en la solicitud
   void&  data[]       // enlace a la variable para obtener el valor
   );
```

Parámetros

request

[in] Manejador de solicitud recibido en DatabasePrepare().

column

[in] Número de campo en la solicitud. La numeración comienza desde cero y no puede superar el valor DatabaseColumnsCount() - 1.

data[]

[out] Enlace a la matriz para registrar el valor del campo.

Valor retornado

Retorna true en caso de éxito, o false en caso de error. Para obtener el código del error, use GetLastError(), posibles respuestas:

- ERR_DATABASE_INVALID_HANDLE (5121) - código de manejador no válido;
- ERR_DATABASE_NO_MORE_DATA (5126) - el índice column es mayor que el valor DatabaseColumnsCount() -1.

Observación

El valor solo se puede obtener en el caso de que se haya realizado de forma preliminar para la solicitud request aunque sea una llamada de DatabaseRead().

Para leer el valor de la siguiente entrada, hay que llamar de forma preliminar a DatabaseRead().

Ver también

DatabasePrepare, DatabaseColumnSize, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

Trabajo con DirectX

Las funciones y sombreadores de DirectX 11 han sido diseñados para la visualización 3D directamente en el gráfico de precios.

Para crear un gráfico en tres dimensiones, debemos crear un contexto gráfico (DXContextCreate) con el tamaño de imagen necesario. A continuación, preparamos los búferes de vértices e índices (DXBufferCreate), y creamos los sombreadores de vértices y píxeles (DXShaderCreate). Esto bastará para dar forma al gráfico.

Para el siguiente nivel del gráfico, debemos añadir los parámetros de entrada (DXInputSet), que sirven para transmitir a los sombreadores los parámetros adicionales de dibujado. Esto nos permitirá establecer la posición de la cámara y el objeto 3D, describir las fuentes de color e implementar el control con el ratón y el teclado.

De esta forma, las funciones incorporadas directamente en MQL5 nos permitirán crear gráficos 3D animados directamente en el terminal MetaTrader 5, sin la ayuda de herramientas de terceros. Para trabajar con las funciones, la tarjeta gráfica deberá soportar DX 11 y los sombreadores de la versión 5.0.

Para comenzar a trabajar con la bilioteca, basta con leer el artículo Cómo crear gráficos 3D en DirectX en MetaTrader 5.

Función	Acción
DXContextCreate	Crea un contexto gráfico para dibujar frames del tamaño indicado
DXContextSetSize	Modifica el tamaño del frame de un contexto gráfico creado en DXContextCreate()
DXContextSetSize	Obtiene el tamaño del frame de un contexto gráfico creado en DXContextCreate()
DXContextClearColors	Establece para el búfer de dibujado todos los píxeles en el color indicado
DXContextClearDepth	Limpia el búfer de profundidad
DXContextGetColors	Obtiene del contexto gráfico la imagen con el tamaño y desplazamiento indicados
DXContextGetDepth	Obtiene el búfer de profundidad del frame dibujado
DXBufferCreate	Crea un búfer del tipo indicado basado en una matriz de datos
DXTextureCreate	Crea una textura en 2 dimensiones a partir de un rectángulo del tamaño indicado, recortado de la imagen transmitida
DXInputCreate	Crea los parámetros de entrada del sombreador
DXInputSet	Establece los parámetros de entrada del sombreador
DXShaderCreate	Crea un sombreador del tipo indicado
DXShaderSetLayout	Establece una marca de vértice para el sombreador de vértices

© 2000-2025, MetaQuotes Ltd.

Función	Acción
DXShaderInputsSet	Establece los parámetros de entrada del sombreador
DXShaderTexturesSet	Establece las texturas para el sombreador
DXDraw	Dibuja los vértices del búfer de vértices establecido en DXBufferSet()
DXDrawIndexed	Dibuja las primitivas gráficas descritas por el búfer de índices de DXBufferSet()
DXPrimiveTopologySet	Establece el tipo de primitivas para el dibujado con la ayuda de DXDrawIndexed()
DXBufferSet	Establece el búfer para el dibujado actual
DXShaderSet	Establece el sombreador para el dibujado
DXHandleType	Retorna el tipo de manejador
DXRelease	Libera el manejador

© 2000-2025, MetaQuotes Ltd.

DXContextCreate

Crea un contexto gráfico para dibujar frames del tamaño indicado.

```
int  DXContextCreate(
   uint  width,      // anchura en píxeles
   uint  height      // altura en píxeles
   );
```

Parámetros

width

[in] Anchura del frame en píxeles.

height

[in] Altura del frame en píxeles

Valor retornado

Manejador del contexto creado o INVALID_HANDLE en el caso de error. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

Todos los objetos gráficos creados con la ayuda de las funciones DXBufferCreate, DXInputCreate, DXShaderCreate y DXTextureCreate, se pueden utilizar solo en el contexto gráfico en el que han sido creados.

Por consiguiente, el tamaño del frame se puede modificar en DXContextSetSize().

Un manejador creado que ya no vaya a usarse más, deberá ser explícitamente liberado con la ayuda de la función DXRelease().

© 2000-2025, MetaQuotes Ltd.

DXContextSetSize

Modifica el tamaño del frame de un contexto gráfico creado en DXContextCreate().

```
bool  DXContextSetSize(
   int    context,      // manejador para el contexto gráfico
   uint&  width,        // anchura en píxeles
   uint&  height        // altura en píxeles
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

width

[in] Anchura del frame en píxeles.

height

[in] Altura del frame en píxeles.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

El tamaño del frame del contexto gráfico solo se modificará entre un dibujado de frame y otro.

© 2000-2025, MetaQuotes Ltd.

DXContextGetSize

Obtiene el tamaño del frame de un contexto gráfico creado en DXContextCreate().

```
bool  DXContextGetSize(
   int    context,      // manejador para el contexto gráfico
   uint&  width,        // anchura en píxeles
   uint&  height        // altura en píxeles
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

width

[out] Anchura del frame en píxeles.

height

[out] Altura del frame en píxeles.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

DXContextClearColors

Establece para el búfer de dibujado todos los píxeles en el color indicado.

```
bool  DXContextClearColors(
   int              context,      // manejador para el contexto gráfico
   const DXVector&  color         // color
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

color

[in] Color para el dibujado.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

La función DXContextClearColors() puede usarse para limpiar el búfer de color antes de dibujar el siguiente frame.

© 2000-2025, MetaQuotes Ltd.

DXContextClearDepth

Limpia el búfer de profundidad.

```
bool  DXContextClearDepth(
   int  context      // manejador para el contexto gráfico
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

La función DXContextClearDepth() puede usarse para limpiar el búfer de profundidad antes de dibujar el siguiente frame.

© 2000-2025, MetaQuotes Ltd.

DXContextGetColors

Obtiene del contexto gráfico la imagen con el tamaño y desplazamiento indicados.

```
bool  DXContextGetColors(
   int    context,                    // manejador para el contexto gráfico
   uint&  image[],                    // matriz de píxeles de la imagen
   int    image_width=WHOLE_ARRAY,    // anchura de la imagen en píxeles
   int    image_height=WHOLE_ARRAY,   // altura de la imagen en píxeles
   int    image_offset_x=0,           // desplazamiento en X
   int    image_offset_y=0            // desplazamiento en Y
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

image

[out] Matriz de píxeles de tamaño *image_width*image_height* en el formato ARGB.

image_width=WHOLE_ARRAY

[in] Anchura de la imagen en píxeles.

image_height=WHOLE_ARRAY

[in] Altura de la imagen en píxeles.

image_offset_x=0

[in] Desplazamiento horizontal.

image_offset_y=0

[in] Desplazamiento vertical.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

DXContextGetDepth

Obtiene el búfer de profundidad del frame dibujado.

```
bool  DXContextGetDepth(
   int     context,      // manejador para el contexto gráfico
   float&  image[]       // matriz de valores de profundidad
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

image

[out] Matriz de valores del búfer de profundidad del frame dibujado.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

El búfer retornado contiene en unidades relativas (de 0.0 a 1.0) la profundidad de cada píxel del frame dibujado que puede ser obtenido en DXContextGetColors().

© 2000-2025, MetaQuotes Ltd.

DXBufferCreate

Crea un búfer del tipo indicado basado en una matriz de datos.

```
int  DXBufferCreate(
   int                  context,              // manejador para el contexto gráfico
   ENUM_DX_BUFFER_TYPE  buffer_type,          // tipo de búfer creado
   const void&          data[],               // datos para el búfer
   uint                 start=0,              // índice inicial
   uint                 count=WHOLE_ARRAY     // número de elementos
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

buffer_type

[in] Tipo de búfer de la enumeración ENUM_DX_BUFFER_TYPE.

data[]

[in] Datos para crear el búfer.

start

[in] Índice del primer elemento de la matriz, a partir del cual se toma el valor de la matriz para crear el búfer. Por defecto, se toman los datos del inicio de la matriz.

count

[in] Número de valores. Por defecto, se usa toda la matriz (count=WHOLE_ARRAY).

Valor retornado

Manejador del búfer creado o INVALID_HANDLE en el caso de error. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

Para el búfer de índices, la matriz *data[]* debe tener el tipo uint, mientras que para el búfer de vértices se transmite una matriz de estructuras que describe los vértices.

Un manejador creado que ya no vaya a usarse más, deberá ser explícitamente liberado con la ayuda de la función DXRelease().

ENUM_DX_BUFFER_TYPE

Identificador	Valor	Descripción
DX_BUFFER_VERTEX	1	Búfer de vértices
DX_BUFFER_INDEX	2	Búfer de índices

© 2000-2025, MetaQuotes Ltd.

DXTextureCreate

Crea una textura en 2 dimensiones a partir de un rectángulo del tamaño indicado, recortado de la imagen transmitida.

```
int  DXTextureCreate(
   int             context,         // manejador para el contexto gráfico
   ENUM_DX_FORMAT  format,          // formato de color del píxel
   uint            width,           // anchura de la imagen original
   uint            height,          // altura de la imagen original
   const void&     data[],          // matriz de píxeles de la imagen original
   uint            data_x,          // coordenada X del rectángulo para crear la textu
   uint            data_y,          // coordenada Y del rectángulo para crear la textu
   uint            data_width,      // anchura del rectángulo para crear la textura
   uint            data_height      // altura del rectángulo para crear la textura
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

format

[in] Formato de color del píxel establecido desde la enumeración ENUM_DX_FORMAT.

width

[in] Anchura de la imagen en la que se basa la textura.

height

[in] Altura de la imagen en la que se basa la textura.

data

[in] Matriz de píxeles de la imagen en la que se basa la textura.

data_x

[in] Coordenada X del rectángulo (desplazamiento horizontal) a partir del cual se crea la textura.

data_y

[in] Coordenada Y del rectángulo (desplazamiento vertical) a partir del cual se crea la textura.

data_width

[in] Anchura del rectángulo a partir del cual se crea la textura.

data_height

[in] Altura del rectángulo a partir del cual se crea la textura.

Valor retornado

Manejador para la textura o INVALID_HANDLE en el caso de error. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

© 2000-2025, MetaQuotes Ltd.

Un manejador creado que ya no vaya a usarse más, deberá ser explícitamente liberado con la ayuda de la función DXRelease().

ENUM_DX_FORMAT

Identificador	Valor	Correspondencia en DXGI_FORMAT
DX_FORMAT_UNKNOWN	0	DXGI_FORMAT_UNKNOWN
DX_FORMAT_R32G32B32A32_TYPELESS	1	DXGI_FORMAT_R32G32B32A32_TYPELESS
DX_FORMAT_R32G32B32A32_FLOAT	2	DXGI_FORMAT_R32G32B32A32_FLOAT
DX_FORMAT_R32G32B32A32_UINT	3	DXGI_FORMAT_R32G32B32A32_UINT
DX_FORMAT_R32G32B32A32_SINT	4	DXGI_FORMAT_R32G32B32A32_SINT
DX_FORMAT_R32G32B32_TYPELESS	5	DXGI_FORMAT_R32G32B32_TYPELESS
DX_FORMAT_R32G32B32_FLOAT	6	DXGI_FORMAT_R32G32B32_FLOAT
DX_FORMAT_R32G32B32_UINT	7	DXGI_FORMAT_R32G32B32_UINT
DX_FORMAT_R32G32B32_SINT	8	DXGI_FORMAT_R32G32B32_SINT
DX_FORMAT_R16G16B16A16_TYPELESS	9	DXGI_FORMAT_R16G16B16A16_TYPELESS
DX_FORMAT_R16G16B16A16_FLOAT	10	DXGI_FORMAT_R16G16B16A16_FLOAT
DX_FORMAT_R16G16B16A16_UNORM	11	DXGI_FORMAT_R16G16B16A16_UNORM
DX_FORMAT_R16G16B16A16_UINT	12	DXGI_FORMAT_R16G16B16A16_UINT
DX_FORMAT_R16G16B16A16_SNORM	13	DXGI_FORMAT_R16G16B16A16_SNORM
DX_FORMAT_R16G16B16A16_SINT	14	DXGI_FORMAT_R16G16B16A16_SINT
DX_FORMAT_R32G32_TYPELESS	15	DXGI_FORMAT_R32G32_TYPELESS
DX_FORMAT_R32G32_FLOAT	16	DXGI_FORMAT_R32G32_FLOAT
DX_FORMAT_R32G32_UINT	17	DXGI_FORMAT_R32G32_UINT
DX_FORMAT_R32G32_SINT	18	DXGI_FORMAT_R32G32_SINT
DX_FORMAT_R32G8X24_TYPELESS	19	DXGI_FORMAT_R32G8X24_TYPELESS
DX_FORMAT_D32_FLOAT_S8X24_UINT	20	DXGI_FORMAT_D32_FLOAT_S8X24_UINT
DX_FORMAT_R32_FLOAT_X8X24_TYPELESS	21	DXGI_FORMAT_R32_FLOAT_X8X24_TYPELESS
DX_FORMAT_X32_TYPELESS_G8X24_UINT	22	DXGI_FORMAT_X32_TYPELESS_G8X24_UINT

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor	Correspondencia en DXGI_FORMAT
DX_FORMAT_R10G10B10A2_TYPELESS	23	DXGI_FORMAT_R10G10B10A2_TYPELESS
DX_FORMAT_R10G10B10A2_UNORM	24	DXGI_FORMAT_R10G10B10A2_UNORM
DX_FORMAT_R10G10B10A2_UINT	25	DXGI_FORMAT_R10G10B10A2_UINT
DX_FORMAT_R11G11B10_FLOAT	26	DXGI_FORMAT_R11G11B10_FLOAT
DX_FORMAT_R8G8B8A8_TYPELESS	27	DXGI_FORMAT_R8G8B8A8_TYPELESS
DX_FORMAT_R8G8B8A8_UNORM	28	DXGI_FORMAT_R8G8B8A8_UNORM
DX_FORMAT_R8G8B8A8_UNORM_SRGB	29	DXGI_FORMAT_R8G8B8A8_UNORM_SRGB
DX_FORMAT_R8G8B8A8_UINT	30	DXGI_FORMAT_R8G8B8A8_UINT
DX_FORMAT_R8G8B8A8_SNORM	31	DXGI_FORMAT_R8G8B8A8_SNORM
DX_FORMAT_R8G8B8A8_SINT	32	DXGI_FORMAT_R8G8B8A8_SINT
DX_FORMAT_R16G16_TYPELESS	33	DXGI_FORMAT_R16G16_TYPELESS
DX_FORMAT_R16G16_FLOAT	34	DXGI_FORMAT_R16G16_FLOAT
DX_FORMAT_R16G16_UNORM	35	DXGI_FORMAT_R16G16_UNORM
DX_FORMAT_R16G16_UINT	36	DXGI_FORMAT_R16G16_UINT
DX_FORMAT_R16G16_SNORM	37	DXGI_FORMAT_R16G16_SNORM
DX_FORMAT_R16G16_SINT	38	DXGI_FORMAT_R16G16_SINT
DX_FORMAT_R32_TYPELESS	39	DXGI_FORMAT_R32_TYPELESS
DX_FORMAT_D32_FLOAT	40	DXGI_FORMAT_D32_FLOAT
DX_FORMAT_R32_FLOAT	41	DXGI_FORMAT_R32_FLOAT
DX_FORMAT_R32_UINT	42	DXGI_FORMAT_R32_UINT
DX_FORMAT_R32_SINT	43	DXGI_FORMAT_R32_SINT
DX_FORMAT_R24G8_TYPELESS	44	DXGI_FORMAT_R24G8_TYPELESS
DX_FORMAT_D24_UNORM_S8_UINT	45	DXGI_FORMAT_D24_UNORM_S8_UINT
DX_FORMAT_R24_UNORM_X8_TYPELESS	46	DXGI_FORMAT_R24_UNORM_X8_TYPELESS
DX_FORMAT_X24_TYPELESS_G8_UINT	47	DXGI_FORMAT_X24_TYPELESS_G8_UINT
DX_FORMAT_R8G8_TYPELESS	48	DXGI_FORMAT_R8G8_TYPELESS
DX_FORMAT_R8G8_UNORM	49	DXGI_FORMAT_R8G8_UNORM
DX_FORMAT_R8G8_UINT	50	DXGI_FORMAT_R8G8_UINT

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor	Correspondencia en DXGI_FORMAT
DX_FORMAT_R8G8_SNORM	51	DXGI_FORMAT_R8G8_SNORM
DX_FORMAT_R8G8_SINT	52	DXGI_FORMAT_R8G8_SINT
DX_FORMAT_R16_TYPELESS	53	DXGI_FORMAT_R16_TYPELESS
DX_FORMAT_R16_FLOAT	54	DXGI_FORMAT_R16_FLOAT
DX_FORMAT_D16_UNORM	55	DXGI_FORMAT_D16_UNORM
DX_FORMAT_R16_UNORM	56	DXGI_FORMAT_R16_UNORM
DX_FORMAT_R16_UINT	57	DXGI_FORMAT_R16_UINT
DX_FORMAT_R16_SNORM	58	DXGI_FORMAT_R16_SNORM
DX_FORMAT_R16_SINT	59	DXGI_FORMAT_R16_SINT
DX_FORMAT_R8_TYPELESS	60	DXGI_FORMAT_R8_TYPELESS
DX_FORMAT_R8_UNORM	61	DXGI_FORMAT_R8_UNORM
DX_FORMAT_R8_UINT	62	DXGI_FORMAT_R8_UINT
DX_FORMAT_R8_SNORM	63	DXGI_FORMAT_R8_SNORM
DX_FORMAT_R8_SINT	64	DXGI_FORMAT_R8_SINT
DX_FORMAT_A8_UNORM	65	DXGI_FORMAT_A8_UNORM
DX_FORMAT_R1_UNORM	66	DXGI_FORMAT_R1_UNORM
DX_FORMAT_R9G9B9E5_SHAREDEXP	67	DXGI_FORMAT_R9G9B9E5_SHAREDEXP
DX_FORMAT_R8G8_B8G8_UNORM	68	DXGI_FORMAT_R8G8_B8G8_UNORM
DX_FORMAT_G8R8_G8B8_UNORM	69	DXGI_FORMAT_G8R8_G8B8_UNORM
DX_FORMAT_BC1_TYPELESS	70	DXGI_FORMAT_BC1_TYPELESS
DX_FORMAT_BC1_UNORM	71	DXGI_FORMAT_BC1_UNORM
DX_FORMAT_BC1_UNORM_SRGB	72	DXGI_FORMAT_BC1_UNORM_SRGB
DX_FORMAT_BC2_TYPELESS	73	DXGI_FORMAT_BC2_TYPELESS
DX_FORMAT_BC2_UNORM	74	DXGI_FORMAT_BC2_UNORM
DX_FORMAT_BC2_UNORM_SRGB	75	DXGI_FORMAT_BC2_UNORM_SRGB
DX_FORMAT_BC3_TYPELESS	76	DXGI_FORMAT_BC3_TYPELESS
DX_FORMAT_BC3_UNORM	77	DXGI_FORMAT_BC3_UNORM
DX_FORMAT_BC3_UNORM_SRGB	78	DXGI_FORMAT_BC3_UNORM_SRGB
DX_FORMAT_BC4_TYPELESS	79	DXGI_FORMAT_BC4_TYPELESS
DX_FORMAT_BC4_UNORM	80	DXGI_FORMAT_BC4_UNORM
DX_FORMAT_BC4_SNORM	81	DXGI_FORMAT_BC4_SNORM

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor	Correspondencia en DXGI_FORMAT
DX_FORMAT_BC5_TYPELESS	82	DXGI_FORMAT_BC5_TYPELESS
DX_FORMAT_BC5_UNORM	83	DXGI_FORMAT_BC5_UNORM
DX_FORMAT_BC5_SNORM	84	DXGI_FORMAT_BC5_SNORM
DX_FORMAT_B5G6R5_UNORM	85	DXGI_FORMAT_B5G6R5_UNORM
DX_FORMAT_B5G5R5A1_UNORM	86	DXGI_FORMAT_B5G5R5A1_UNORM
DX_FORMAT_B8G8R8A8_UNORM	87	DXGI_FORMAT_B8G8R8A8_UNORM
DX_FORMAT_B8G8R8X8_UNORM	88	DXGI_FORMAT_B8G8R8X8_UNORM
DX_FORMAT_R10G10B10_XR_BIAS_A2_UNORM	89	DXGI_FORMAT_R10G10B10_XR_BIAS_A2_UNORM
DX_FORMAT_B8G8R8A8_TYPELESS	90	DXGI_FORMAT_B8G8R8A8_TYPELESS
DX_FORMAT_B8G8R8A8_UNORM_SRGB	91	DXGI_FORMAT_B8G8R8A8_UNORM_SRGB
DX_FORMAT_B8G8R8X8_TYPELESS	92	DXGI_FORMAT_B8G8R8X8_TYPELESS
DX_FORMAT_B8G8R8X8_UNORM_SRGB	93	DXGI_FORMAT_B8G8R8X8_UNORM_SRGB
DX_FORMAT_BC6H_TYPELESS	94	DXGI_FORMAT_BC6H_TYPELESS
DX_FORMAT_BC6H_UF16	95	DXGI_FORMAT_BC6H_UF16
DX_FORMAT_BC6H_SF16	96	DXGI_FORMAT_BC6H_SF16
DX_FORMAT_BC7_TYPELESS	97	DXGI_FORMAT_BC7_TYPELESS
DX_FORMAT_BC7_UNORM	98	DXGI_FORMAT_BC7_UNORM
DX_FORMAT_BC7_UNORM_SRGB	99	DXGI_FORMAT_BC7_UNORM_SRGB
DX_FORMAT_AYUV	100	DXGI_FORMAT_AYUV
DX_FORMAT_Y410	101	DXGI_FORMAT_Y410
DX_FORMAT_Y416	102	DXGI_FORMAT_Y416
DX_FORMAT_NV12	103	DXGI_FORMAT_NV12
DX_FORMAT_P010	104	DXGI_FORMAT_P010
DX_FORMAT_P016	105	DXGI_FORMAT_P016
DX_FORMAT_420_OPAQUE	106	DXGI_FORMAT_420_OPAQUE
DX_FORMAT_YUY2	107	DXGI_FORMAT_YUY2
DX_FORMAT_Y210	108	DXGI_FORMAT_Y210
DX_FORMAT_Y216	109	DXGI_FORMAT_Y216
DX_FORMAT_NV11	110	DXGI_FORMAT_NV11

© 2000-2025, MetaQuotes Ltd.

Identificador	Valor	Correspondencia en DXGI_FORMAT
DX_FORMAT_AI44	111	DXGI_FORMAT_AI44
DX_FORMAT_IA44	112	DXGI_FORMAT_IA44
DX_FORMAT_P8	113	DXGI_FORMAT_P8
DX_FORMAT_A8P8	114	DXGI_FORMAT_A8P8
DX_FORMAT_B4G4R4A4_UNORM	115	DXGI_FORMAT_B4G4R4A4_UNORM
DX_FORMAT_P208	130	DXGI_FORMAT_P208
DX_FORMAT_V208	131	DXGI_FORMAT_V208
DX_FORMAT_V408	132	DXGI_FORMAT_V408
DX_FORMAT_FORCE_UINT	0xffffffff	DXGI_FORMAT_FORCE_UINT

© 2000-2025, MetaQuotes Ltd.

DXInputCreate

Crea los parámetros de entrada del sombreador.

```
int  DXInputCreate(
   int   context,        // manejador para el contexto gráfico
   uint  input_size      // tamaño de los parámetros de entrada en bytes
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

input_size

[in] Tamaño de la estructura de parámetros en bytes.

Valor retornado

Manejador para los parámetros de entrada del sombreador o INVALID_HANDLE en el caso de error. Para obtener el código del error, deberemos llamar a la función GetLastError().

Un manejador creado que ya no vaya a usarse más, deberá ser explícitamente liberado con la ayuda de la función DXRelease().

© 2000-2025, MetaQuotes Ltd.

DXInputSet

Establece los parámetros de entrada del sombreador.

```
bool  DXInputSet(
   int           input,      // manejador para el contexto gráfico
   const void&   data        // datos para establecer
   );
```

Parámetros

input

[in] Manejador para los parámetros de entrada del sombreador obtenido en DXInputCreate().

data

[in] Datos para establecer los parámetros de entrada del sombreador.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

© 2000-2025, MetaQuotes Ltd.

DXShaderCreate

Crea un sombreador del tipo indicado.

```
int  DXShaderCreate(
   int                  context,          // manejador para el contexto gráfico
   ENUM_DX_SHADER_TYPE  shader_type,      // tipo de sombreado
   const string         source,           // código fuente del sombreador
   const string         entry_point,      // punto de entrada
   string&              compile_error     // línea para obtener los mensajes del compi
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

shader_type

[out] Valor de la enumeración ENUM_DX_SHADER_TYPE.

source

[in] Código fuente del sombreador escrito en HLSL 5.

entry_point

[in] Punto de entrada - nombre de la función en el código fuente.

compile_error

[in] Línea para obtener los errores de compilación.

Valor retornado

Manejador para el sombreador o INVALID_HANDLE en el caso de error. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

Un manejador creado que ya no vaya a usarse más, deberá ser explícitamente liberado con la ayuda de la función DXRelease().

ENUM_DX_SHADER_TYPE

Identificador	Valor	Descripción
DX_SHADER_VERTEX	0	Sombreador de vértices
DX_SHADER_GEOMETRY	1	Sombreador geométrico
DX_SHADER_PIXEL	2	Sombreador de píxeles

© 2000-2025, MetaQuotes Ltd.

DXShaderSetLayout

Establece una marca de vértice para el sombreador de vértices.

```
bool  DXShaderSetLayout(
   int                     shader,       // manejador del sombreador
   const DXVertexLayout&   layout[]      //
   );
```

Parámetros

shader

[in] Manejador del búfer de vértices creado en DXShaderCreate().

layout[]

[in] Matriz de descripción para los campos de los vértices. La descripción se realiza con la estructura DXVertexLayout:

```
struct DXVertexLayout
  {
   string         semantic_name;        // The HLSL semantic associated with this e
   uint           semantic_index;       // The semantic index for the element. A se
   ENUM_DX_FORMAT format;               // The data type of the element data.
  };
```

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

La disposición deberá corresponderse con el tipo de vértices en el búfer de vértices establecido, así como con el tipo de entrada de vértices usado en el punto de entrada en el código del sombreador de vértices.

El búfer de vértices para el sombreador se establece en DXBufferSet().

La estructura DXVertexLayout es una versión de la estructura D3D11_INPUT_ELEMENT_DESC de MSDN.

© 2000-2025, MetaQuotes Ltd.

DXShaderInputsSet

Establece los parámetros de entrada del sombreador.

```
bool  DXShaderInputsSet(
   int          shader,        // manejador para el sombreador
   const int&   inputs[]       // matriz de manejadores de los parámetros de entrada
   );
```

Parámetros

shader

[in] Manejador del sombreador creado en DXShaderCreate().

inputs[]

[in] Matriz de los manejadores de los parámetros de entrada creados con la ayuda de DXInputCreate().

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

El tamaño de la matriz de los parámetros de entrada debe ser igual al número de objetos cbuffer declarados en el código del sombreador.

© 2000-2025, MetaQuotes Ltd.

DXShaderTexturesSet

Establece las texturas para el sombreador.

```
bool  DXShaderTexturesSet(
   int           shader,          // manejador para el sombreador
   const  int&   textures[]       // matriz de los manejadores de estructuras
   );
```

Parámetros

shader

[in] Manejador del sombreador creado en DXShaderCreate().

textures[]

[in] Matriz de los manejadores de texturas creados con la ayuda de DXTextureCreate().

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

El tamaño de la matriz de texturas debe ser igual al número de objetos Texture2D declarados en el código del sombreador.

© 2000-2025, MetaQuotes Ltd.

DXDraw

Dibuja los vértices del búfer de vértices establecido en DXBufferSet().

```
bool  DXDraw(
   int   context,               // manejador para el contexto gráfico
   uint  start=0,               // índice del primer vértice
   uint  count=WHOLE_ARRAY      // número de vértices
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

start

[in] Índice del primer vértice para el dibujado.

count

[in] Número de vértices para el dibujado.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

Para dibujar los vértices, se deberán establecer preliminarmente los sombreadores con la ayuda de DXShaderSet().

© 2000-2025, MetaQuotes Ltd.

DXDrawIndexed

Dibuja las primitivas gráficas descritas por el búfer de índices de DXBufferSet().

```
bool  DXDrawIndexed(
   int   context,               // manejador para el contexto gráfico
   uint  start=0,               // índice de la primera primitiva
   uint  count=WHOLE_ARRAY      // número de primitivas
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

start

[in] Índice de la primera primitiva para el dibujado.

count

[in] Número de primitivas para el dibujado.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

El tipo de primitivas descritas por el búfer de índices se establece desde DXPrimiveTopologySet().

Para dibujar las primitivas, se deberá establecer preliminarmente el búfer de vértices en DXBufferSet().

Para dibujar las primitivas, se deberán establecer preliminarmente los sombreadores con la ayuda de DXShaderSet().

© 2000-2025, MetaQuotes Ltd.

DXPrimiveTopologySet

Establece el tipo de primitivas para el dibujado con la ayuda de DXDrawIndexed().

```
bool  DXPrimiveTopologySet(
   int                         context,              // manejador para el contexto g
   ENUM_DX_PRIMITIVE_TOPOLOGY  primitive_topology    // tipo de primitiva
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

primitive_topology

[in] Valor de la enumeración ENUM_DX_PRIMITIVE_TOPOLOGY.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

ENUM_DX_PRIMITIVE_TOPOLOGY

Identificador	Valor	Correspondencia en D3D11_PRIMITIVE_TOPOLOGY
DX_PRIMITIVE_TOPOLOGY_POINTLIST	1	D3D11_PRIMITIVE_TOPOLOGY_POINTLIST
DX_PRIMITIVE_TOPOLOGY_LINELIST	2	D3D11_PRIMITIVE_TOPOLOGY_LINELIST
DX_PRIMITIVE_TOPOLOGY_LINESTRIP	3	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST	4	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP	5	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP
DX_PRIMITIVE_TOPOLOGY_LINELIST_ADJ	6	D3D11_PRIMITIVE_TOPOLOGY_LINELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ	7	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ	8	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ	9	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ

© 2000-2025, MetaQuotes Ltd.

DXBufferSet

Establece el búfer para el dibujado actual.

```
bool  DXBufferSet(
   int   context,               // manejador para el contexto gráfico
   int   buffer,                // manejador del búfer de vértices o de índices
   uint  start=0,               // índice inicial
   uint  count=WHOLE_ARRAY      // número de elementos
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

buffer

[in] Manejador del búfer de vértices o de índices creado en DXBufferCreate().

start

[in] Índice del primer elemento del búfer. Por defecto, se usan los datos del inicio del búfer.

count

[in] Número de valores que se deben utilizar. Por defecto, todos los valores del búfer.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

Debemos llamar a la función DXBufferSet() para establecer los búferes de vértices e índices para el dibujado con la ayuda de DXDraw().

© 2000-2025, MetaQuotes Ltd.

DXShaderSet

Establece el sombreador para el dibujado.

```
bool  DXShaderSet(
   int  context,      // manejador para el contexto gráfico
   int  shader        // manejador del sombreador
   );
```

Parámetros

context

[in] Manejador del contexto gráfico creado en DXContextCreate().

shader

[in] Manejador del sombreador creado en DXShaderCreate().

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

A la hora de dibujar, se pueden utilizar varios tipos de sombreadores: de vértices, geométricos y de píxeles.

© 2000-2025, MetaQuotes Ltd.

DXHandleType

Retorna el tipo de manejador.

```
ENUM_DX_HANDLE_TYPE  DXHandleType(
   int  handle      // manejador
   );
```

Parámetros

handle

[in] Manejador.

Valor retornado

Valor de la enumeración ENUM_DX_HANDLE_TYPE

ENUM_DX_HANDLE_TYPE

Identificador	Valor	Descripción
DX_HANDLE_INVALID	0	Manejador no válido
DX_HANDLE_CONTEXT	1	Manejador para el contexto gráfico
DX_HANDLE_SHADER	2	Manejador para el sombreador
DX_HANDLE_BUFFER	3	Manejador para el búfer de vértices o de índices
DX_HANDLE_INPUT	4	Manejador para los parámetros de entrada del sombreador
DX_HANDLE_TEXTURE	5	Manejador para la textura

© 2000-2025, MetaQuotes Ltd.

DXRelease

Libera el manejador.

```
bool  DXRelease(
   int  handle      // manejador
   );
```

Parámetros

context

[in] Libera el manejador.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código del error, deberemos llamar a la función GetLastError().

Observación

Todos los manejadores creados que ya no vayan a usarse más, deberán ser explícitamente liberados con la ayuda de la función DXRelease().

© 2000-2025, MetaQuotes Ltd.

Módulo MetaTrader para la integración con Python

MQL5 ha sido pensado para el desarrollo de aplicaciones comerciales de alto rendimiento en los mercados financieros y no tiene análogos entre otros lenguajes especializados, utilizados en el trading algorítmico. La sintaxis y velocidad de funcionamiento de los programas en MQL5 se encuentra muy próxima a C++, dispone de soporte de OpenCL e integración con MS Visual Studio, y también de una biblioteca de estadística, lógica difusa y ALGLIB. En el entorno de desarrollo del MetaEditor también existe el soporte nativo de bibliotecas .NET con importación "inteligente" de funciones sin necesidad de escribir envoltorios especiales, ya que es posible usar DLL de C++ de terceros. Los archivos de los códigos fuente en C++ (CPP y H) se pueden redactar y compilar en DLL directamente desde el editor. Para ello, se usa el Microsoft Visual Studio instalado en la computadora del usuario.

Python es un lenguaje de programación moderno y de alto nivel para el desarrollo de escenarios y aplicaciones. Contiene multitud de bibliotecas para el aprendizaje de máquinas, la automatización de procesos, y también el análisis y visualización de datos.

El paquete MetaTrader para Python ha sido pensado para obtener de forma rápida y sencilla información bursátil a través de la interacción entre procesadores directamente desde el terminal MetaTrader 5. Los datos obtenidos de esta forma se pueden utilizar en lo sucesivo para cálculos estadísticos y aprendizaje de máquinas.

Instalar el paquete en la línea de comandos:

```
pip install MetaTrader5
```

Actualizar el paquete en la línea de comandos:

```
pip install --upgrade MetaTrader5
```

Funciones para la integración de MetaTrader 5 y Python

Función	Acción
initialize	Establece una conexión con el terminal MetaTrader 5
login	Se conecta a la cuenta comercial con los parámetros indicados
shutdown	Cierra una conexión anteriormente establecida con el terminal MetaTrader 5
version	Retorna la versión del terminal MetaTrader 5
last_error	Retorna la información sobre el último error
account_info	Obtiene la información sobre la cuenta comercial actual
terminal_Info	Obtiene el estado y los parámetros del terminal MetaTrader 5 conectado
symbols_total	Obtiene el número total de instrumentos financieros en el terminal MetaTrader 5
symbols_get	Obtiene todos los instrumentos financieros del terminal MetaTrader 5
symbol_info	Obtiene la información sobre el instrumento financiero indicado

© 2000-2025, MetaQuotes Ltd.

Función	Acción
symbol_info_tick	Obtiene el último tick del instrumento financiero indicado
symbol_select	Selecciona un símbolo en la ventana MarketWatch o quita un símbolo de esta ventana
market_book_add	Realiza la suscripción del terminal MetaTrader 5 para recibir eventos sobre los cambios en la profundidad de mercado
market_book_get	Retorna desde BookInfo la tupla que contiene las entradas de la profundidad de mercado del símbolo indicado
market_book_release	Cancela la suscripción del terminal MetaTrader 5 para recibir eventos sobre los cambios en la profundidad de mercado
copy_rates_from	Obtiene las barras del terminal MetaTrader 5, a partir de la fecha indicada
copy_rates_from_pos	Obtiene las barras del terminal MetaTrader 5, a partir del índice establecido
copyrates_range	Obtiene las barras en el intervalo de fechas indicado del terminal MetaTrader 5
copy_ticks_from	Obtiene los ticks del terminal MetaTrader 5, a partir de la fecha indicada
copy_ticks_range	Obtiene los ticks en el intervalo de fechas indicado del terminal MetaTrader 5
orders_total	Obtiene el número de órdenes activas.
orders_get	Obtiene la órdenes activas con posibilidad de filtrado según un símbolo o ticket
order_calc_margin	Retorna el tamaño del margen en la divisa de la cuenta para realizar la operación comercial indicada
order_calc_profit	Retorna el tamaño del beneficio en la divisa de la cuenta para la operación comercial indicada
order_check	Comprueba si los fondos son suficientes para realizar la operación comercial requerida
order_send	Envía desde el terminal al servidor comercial una solicitud de ejecución de una operación comercial.
positions_total	Obtiene el número de posiciones abiertas
positions_get	Obtiene las posiciones abiertas con posibilidad de filtrado según un símbolo o ticket
history_orders_total	Obtiene el número de órdenes en la historia comercial en el intervalo indicado
history_orders_get	Obtiene las órdenes de la historia comercial con posibilidad de filtrado según un ticket o posición

© 2000-2025, MetaQuotes Ltd.

Función	Acción
history_deals_total	Obtiene el número de transacciones en la historia comercial en el intervalo indicado
history_deals_get	Obtiene las transacciones de la historia comercial con posibilidad de filtrado según un ticket o posición

Ejemplo de conexión de Python a MetaTrader 5

1. Descargue la última versión Python 3.8 de la página https://www.python.org/downloads/windows
2. Al instalar Python, marque la casilla de verificación "Add Python 3.8 to PATH%", para que sea posible iniciar scripts en Python desde la línea de comandos.
3. Instale el módulo MetaTrader5 desde la línea de comandos

```
pip install MetaTrader5
```

4. Añada los paquetes matplotlib y pandas

```
pip install matplotlib
pip install pandas
```

5. Inicie el script de prueba

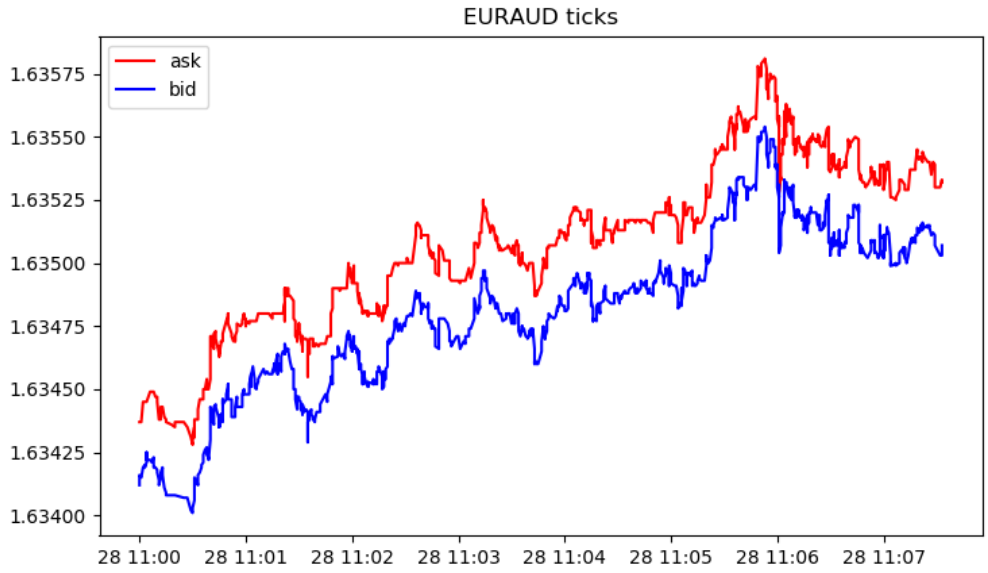
```
from datetime import datetime
import matplotlib.pyplot as plt
import pandas as pd
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import MetaTrader5 as mt5

# conectamos con MetaTrader 5
if not mt5.initialize():
    print("initialize() failed")
    mt5.shutdown()

# solicitamos el estado y los parámetros de conexión
print(mt5.terminal_info())
# obtenemos la información sobre la versión de MetaTrader 5
print(mt5.version())

# solicitamos 1000 ticks de EURAUD
euraud_ticks = mt5.copy_ticks_from("EURAUD", datetime(2020,1,28,13), 1000, mt5.COPY_TI
# solicitamos los ticks de AUDUSD en el intervalo 2019.04.01 13:00 - 2019.04.02 13:00
audusd_ticks = mt5.copy_ticks_range("AUDUSD", datetime(2020,1,27,13), datetime(2020,1,

# obtenemos con distintos métodos las barras de diferentes instrumentos
eurusd_rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_M1, datetime(2020,1,28,13),
eurgbp_rates = mt5.copy_rates_from_pos("EURGBP", mt5.TIMEFRAME_M1, 0, 1000)
eurcad_rates = mt5.copy_rates_range("EURCAD", mt5.TIMEFRAME_M1, datetime(2020,1,27,13)

# finalizamos la conexión con MetaTrader 5
```

© 2000-2025, MetaQuotes Ltd.

```
mt5.shutdown()

#DATA
print('euraud_ticks(', len(euraud_ticks), ')')
for val in euraud_ticks[:10]: print(val)

print('audusd_ticks(', len(audusd_ticks), ')')
for val in audusd_ticks[:10]: print(val)

print('eurusd_rates(', len(eurusd_rates), ')')
for val in eurusd_rates[:10]: print(val)

print('eurgbp_rates(', len(eurgbp_rates), ')')
for val in eurgbp_rates[:10]: print(val)

print('eurcad_rates(', len(eurcad_rates), ')')
for val in eurcad_rates[:10]: print(val)

#PLOT
# creamos un DataFrame de los datos obtenidos
ticks_frame = pd.DataFrame(euraud_ticks)
# dibujamos los ticks en el gráfico
plt.plot(ticks_frame['time'], ticks_frame['ask'], 'r-', label='ask')
plt.plot(ticks_frame['time'], ticks_frame['bid'], 'b-', label='bid')

# mostramos los rótulos
plt.legend(loc='upper left')

# añadimos los encabezados
plt.title('EURAUD ticks')

# mostramos el gráfico
plt.show()
```

© 2000-2025, MetaQuotes Ltd.

6. Obtenga los datos y el gráfico




```
[2, 'MetaQuotes-Demo', '16167573']
[500, 2325, '19 Feb 2020']

euraud_ticks( 1000 )
(1580209200, 1.63412, 1.63437, 0., 0, 1580209200067, 130, 0.)
(1580209200, 1.63416, 1.63437, 0., 0, 1580209200785, 130, 0.)
(1580209201, 1.63415, 1.63437, 0., 0, 1580209201980, 130, 0.)
(1580209202, 1.63419, 1.63445, 0., 0, 1580209202192, 134, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203004, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203487, 130, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203694, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203990, 130, 0.)
(1580209204, 1.63421, 1.63445, 0., 0, 1580209204194, 130, 0.)
(1580209204, 1.63425, 1.63445, 0., 0, 1580209204392, 130, 0.)
audusd_ticks( 40449 )
(1580122800, 0.67858, 0.67868, 0., 0, 1580122800244, 130, 0.)
(1580122800, 0.67858, 0.67867, 0., 0, 1580122800429, 4, 0.)
(1580122800, 0.67858, 0.67865, 0., 0, 1580122800817, 4, 0.)
(1580122801, 0.67858, 0.67866, 0., 0, 1580122801618, 4, 0.)
(1580122802, 0.67858, 0.67865, 0., 0, 1580122802928, 4, 0.)
(1580122809, 0.67855, 0.67865, 0., 0, 1580122809526, 130, 0.)
(1580122809, 0.67855, 0.67864, 0., 0, 1580122809699, 4, 0.)
(1580122813, 0.67855, 0.67863, 0., 0, 1580122813576, 4, 0.)
(1580122815, 0.67856, 0.67863, 0., 0, 1580122815190, 130, 0.)
(1580122815, 0.67855, 0.67863, 0., 0, 1580122815479, 130, 0.)
eurusd_rates( 1000 )
(1580149260, 1.10132, 1.10151, 1.10131, 1.10149, 44, 1, 0)
(1580149320, 1.10149, 1.10161, 1.10143, 1.10154, 42, 1, 0)
(1580149380, 1.10154, 1.10176, 1.10154, 1.10174, 40, 2, 0)
(1580149440, 1.10174, 1.10189, 1.10168, 1.10187, 47, 1, 0)
```

© 2000-2025, MetaQuotes Ltd.

```
(1580149500, 1.10185, 1.10191, 1.1018, 1.10182, 53, 1, 0)
(1580149560, 1.10182, 1.10184, 1.10176, 1.10183, 25, 3, 0)
(1580149620, 1.10183, 1.10187, 1.10177, 1.10187, 49, 2, 0)
(1580149680, 1.10187, 1.1019, 1.1018, 1.10187, 53, 1, 0)
(1580149740, 1.10187, 1.10202, 1.10187, 1.10198, 28, 2, 0)
(1580149800, 1.10198, 1.10198, 1.10183, 1.10188, 39, 2, 0)
eurgbp_rates( 1000 )
(1582236360, 0.83767, 0.83767, 0.83764, 0.83765, 23, 9, 0)
(1582236420, 0.83765, 0.83765, 0.83764, 0.83765, 15, 8, 0)
(1582236480, 0.83765, 0.83766, 0.83762, 0.83765, 19, 7, 0)
(1582236540, 0.83765, 0.83768, 0.83758, 0.83763, 39, 6, 0)
(1582236600, 0.83763, 0.83768, 0.83763, 0.83767, 21, 6, 0)
(1582236660, 0.83767, 0.83775, 0.83765, 0.83769, 63, 5, 0)
(1582236720, 0.83769, 0.8377, 0.83758, 0.83764, 40, 7, 0)
(1582236780, 0.83766, 0.83769, 0.8376, 0.83766, 37, 6, 0)
(1582236840, 0.83766, 0.83772, 0.83763, 0.83772, 22, 6, 0)
(1582236900, 0.83772, 0.83773, 0.83768, 0.8377, 36, 5, 0)
eurcad_rates( 1441 )
(1580122800, 1.45321, 1.45329, 1.4526, 1.4528, 146, 15, 0)
(1580122860, 1.4528, 1.45315, 1.45274, 1.45301, 93, 15, 0)
(1580122920, 1.453, 1.45304, 1.45264, 1.45264, 82, 15, 0)
(1580122980, 1.45263, 1.45279, 1.45231, 1.45277, 109, 15, 0)
(1580123040, 1.45275, 1.4528, 1.45259, 1.45271, 53, 14, 0)
(1580123100, 1.45273, 1.45285, 1.45269, 1.4528, 62, 16, 0)
(1580123160, 1.4528, 1.45284, 1.45267, 1.45282, 64, 14, 0)
(1580123220, 1.45282, 1.45299, 1.45261, 1.45272, 48, 14, 0)
(1580123280, 1.45272, 1.45275, 1.45255, 1.45275, 74, 14, 0)
(1580123340, 1.45275, 1.4528, 1.4526, 1.4528, 94, 13, 0)
```

© 2000-2025, MetaQuotes Ltd.

initialize

Establece una conexión con el terminal MetaTrader 5. Existen 3 variantes de llamada.

Llamada sin parámetros. El terminal para la conexión se encontrará de forma automática.

```
initialize()
```

Llamada con indicación de la ruta hasta el terminal MetaTrader 5 al que debemos conectarnos.

```
initialize(
   path                       // ruta al archivo EXE del terminal MetaTrader 5
   )
```

Llamada con indicación de la ruta y los parámetros de la cuenta comercial.

```
initialize(
   path,                      // ruta al archivo EXE del terminal MetaTrader 5
   login=LOGIN,               // número de cuenta
   password="PASSWORD",       // contraseña
   server="SERVER",           // nombre del servidor como se ha establecido en el termi
   timeout=TIMEOUT,           // timeout
   portable=False             // modo portable
   )
```

Parámetros

path

[in] Ruta al archivo metatrader.exe o metatrader64.exe. Parámetro no nombrado no obligatorio. Se indica en primer lugar, sin el nombre del parámetro. Si no ha sido indicado, el módulo intentará encontrar el archivo ejecutable por sí mismo.

login=LOGIN

[in] Número de la cuenta comercial. Parámetro nombrado no obligatorio. Si no se indica, se usará la última cuenta comercial.

password="PASSWORD"

[in] Contraseña para la cuenta comercial. Parámetro nombrado no obligatorio. Si no se indica la contraseña, se utilizará automáticamente la contraseña guardada en la base de datos del terminal para la cuenta indicada.

server="SERVER"

[in] Nombre del servidor comercial. Parámetro nombrado no obligatorio. Si no se indica el servidor, se utilizará automáticamente el servidor guardado en la base de datos del terminal para la cuenta indicada.

timeout=TIMEOUT

[in] Timeout en milisegundos disponible para la conexión. Parámetro nombrado no obligatorio. Si no se indica, se usará el valor 60000 (60 segundos).

portable=False

[in] Señal de inicio del terminal en el modo portable. Parámetro nombrado no obligatorio. Si no se indica, se usará el valor False.

© 2000-2025, MetaQuotes Ltd.

Valor retornado

Retorna True si la conexión con el terminal MetaTrader 5 ha tenido éxito, de lo contrario, False.

Observación

Si es necesario, al ejecutar la llamada de initialize(), el terminal MetaTrader 5 se iniciará para ejecutar la conexión.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5 en la cuenta comercial indicad
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# mostramos la información sobre el estado de la conexión, el nombre del servidor y la
print(mt5.terminal_info())
# mostramos la información sobre la versión de MetaTrader 5
print(mt5.version())

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
```

Ver también

shutdown, terminal_info, version

© 2000-2025, MetaQuotes Ltd.

login

Se conecta a la cuenta comercial con los parámetros indicados.

```
login(
   login,                    // número de cuenta
   password="PASSWORD",      // contraseña
   server="SERVER",          // nombre del servidor como se ha establecido en el termi
   timeout=TIMEOUT           // timeout
   )
```

Parámetros

login

[in] Número de la cuenta comercial. Parámetro no nombrado obligatorio.

password

[in] Contraseña para la cuenta comercial. Parámetro nombrado no obligatorio. Si no se indica la contraseña, se utilizará automáticamente la contraseña guardada en la base de datos del terminal.

server

[in] Nombre del servidor comercial. Parámetro nombrado no obligatorio. Si no se indica el servidor, se usará automáticamente el último servidor utilizado.

timeout=TIMEOUT

[in] Timeout en milisegundos disponible para la conexión. Parámetro nombrado no obligatorio. Si no se indica, se usará el valor 60000 (60 segundos). Si no se ha logrado establecer la conexión en este tiempo, la llamada se finalizará forzosamente y se generará una exclusión.

Valor retornado

True si la conexión con la cuenta comercial ha tenido éxito, de lo contrario, False.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# mostramos la información sobre la versión de MetaTrader 5
print(mt5.version())
# conectamos con la cuenta comercial sin indicar la contraseña y el servidor
account=17221085
authorized=mt5.login(account)  # la contraseña se tomará de la base de datos del termi
if authorized:
    print("connected to account #{}".format(account))
else:
```

© 2000-2025, MetaQuotes Ltd.

```
    print("failed to connect at account #{}, error code: {}".format(account, mt5.last_

# ahora, conectamos con la cuenta comercial indicando la contraseña
account=25115284
authorized=mt5.login(account, password="gqrtz0lbdm")
if authorized:
    # mostramos como son los datos sobre la cuenta
    print(mt5.account_info())
    # mostramos los datos sobre la cuenta comercial en forma de lista
    print("Show account_info()._asdict():")
    account_info_dict = mt5.account_info()._asdict()
    for prop in account_info_dict:
        print("  {}={}".format(prop, account_info_dict[prop]))
else:
    print("failed to connect at account #{}, error code: {}".format(account, mt5.last_

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
[500, 2367, '23 Mar 2020']

connected to account #17221085

connected to account #25115284
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
account properties:
   login=25115284
   trade_mode=0
   leverage=100
   limit_orders=200
   margin_so_mode=0
   trade_allowed=True
   trade_expert=True
   margin_mode=2
   currency_digits=2
   fifo_close=False
   balance=99588.33
   credit=0.0
   profit=-45.23
   equity=99543.1
   margin=54.37
   margin_free=99488.73
   margin_level=183084.6054809638
   margin_so_call=50.0
   margin_so_so=30.0
```

© 2000-2025, MetaQuotes Ltd.

```
    margin_initial=0.0
    margin_maintenance=0.0
    assets=0.0
    liabilities=0.0
    commission_blocked=0.0
    name=James Smith
    server=MetaQuotes-Demo
    currency=USD
    company=MetaQuotes Software Corp.
```

Ver también

initialize, shutdown

© 2000-2025, MetaQuotes Ltd.

shutdown

Cierra una conexión anteriormente establecida con el terminal MetaTrader 5.

```
shutdown()
```

Valor retornado

No.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed")
    quit()

# mostramos la información sobre el estado de la conexión, el nombre del servidor y la
print(mt5.terminal_info())
# mostramos la información sobre la versión de MetaTrader 5
print(mt5.version())

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
```

Ver también

initialize, login_py, terminal_info, version

© 2000-2025, MetaQuotes Ltd.

version

Retorna la versión del terminal MetaTrader 5.

```
version()
```

Valor retornado

Retorna el número de versión, el número de build y la fecha de lanzamiento del terminal MetaTrader 5. En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función version() retorna el número de versión, el número de build y la fecha de lanzamiento del terminal en forma de tupla con tres valores:

Tipo	Descripción	Ejemplo de valor
integer	Versión del terminal MetaTrader 5	500
integer	Número de build	2007
string	Fecha de publicación del build	'25 Feb 2019'

Ejemplo:

```
import MetaTrader5 as mt5
import pandas as pd
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# mostramos la información sobre la versión del terminal MetaTrader 5
print(mt5.version())

# mostramos como es la información sobre el estado de la conexión, el nombre del servi
print(mt5.terminal_info())
print()

# obtenemos las propiedades en forma de diccionario
terminal_info_dict=mt5.terminal_info()._asdict()
# transformamos el diccionario en DataFrame y lo imprimimos
df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
print("terminal_info() as dataframe:")
```

© 2000-2025, MetaQuotes Ltd.

```
print(df[:-1])

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
[500, 2367, '23 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True, dlls_a

terminal_info() as dataframe:
                  property                          value
0       community_account                           True
1    community_connection                           True
2               connected                           True
3            dlls_allowed                          False
4           trade_allowed                          False
5       tradeapi_disabled                          False
6           email_enabled                          False
7             ftp_enabled                          False
8   notifications_enabled                          False
9                    mqid                          False
10                  build                           2367
11                maxbars                           5000
12               codepage                           1251
13             ping_last                          77881
14      community_balance                        707.107
15         retransmission                              0
16                company     MetaQuotes Software Corp.
17                   name                  MetaTrader 5
18               language                        Russian
19                   path  E:\ProgramFiles\MetaTrader 5
20              data_path  E:\ProgramFiles\MetaTrader 5
```

Ver también

initialize, shutdown, terminal_info

© 2000-2025, MetaQuotes Ltd.

last_error

Retorna la información sobre el último error.

```
last_error()
```

Valor retornado

Retorna el código del último error y su descripción en forma de tupla.

Observación

last_error() permite obtener el código de error en el caso de que no se ejecute con éxito alguna función de la biblioteca MetaTrader 5. Es un análogo de GetLastError(), pero se usan códigos de error propios. Posibles valores:

Constante	Valor	Descripción
RES_S_OK	1	generic success
RES_E_FAIL	-1	generic fail
RES_E_INVALID_PARAMS	-2	invalid arguments/parameters
RES_E_NO_MEMORY	-3	no memory condition
RES_E_NOT_FOUND	-4	no history
RES_E_INVALID_VERSION	-5	invalid version
RES_E_AUTH_FAILED	-6	authorization failed
RES_E_UNSUPPORTED	-7	unsupported method
RES_E_AUTO_TRADING_DISABLED	-8	auto-trading disabled
RES_E_INTERNAL_FAIL	-10000	internal IPC general error
RES_E_INTERNAL_FAIL_SEND	-10001	internal IPC send failed
RES_E_INTERNAL_FAIL_RECEIVE	-10002	internal IPC recv failed
RES_E_INTERNAL_FAIL_INIT	-10003	internal IPC initialization fail
RES_E_INTERNAL_FAIL_CONNECT	-10003	internal IPC no ipc
RES_E_INTERNAL_FAIL_TIMEOUT	-10005	internal timeout

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2025, MetaQuotes Ltd.

```
    quit()

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
```

Ver también

version, GetLastError

© 2000-2025, MetaQuotes Ltd.

account_info

Obtiene la información sobre la cuenta comercial actual.

```
account_info()
```

Valor retornado

Retorna la información como estructura de tuplas nombradas (namedtuple). En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función retorna en una sola llamada toda la información que se puede obtener con la ayuda de AccountInfoInteger, AccountInfoDouble y AccountInfoString.

Ejemplo:

```
import MetaTrader5 as mt5
import pandas as pd
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# conectamos con la cuenta indicando la contraseña y el servidor
authorized=mt5.login(25115284, password="gqz0343lbdm")
if authorized:
    account_info=mt5.account_info()
    if account_info!=None:
        # mostramos como son los datos sobre la cuenta
        print(account_info)
        # mostramos los datos sobre la cuenta comercial en forma de diccionario
        print("Show account_info()._asdict():")
        account_info_dict = mt5.account_info()._asdict()
        for prop in account_info_dict:
            print("  {}={}".format(prop, account_info_dict[prop]))
        print()

       # transformamos el diccionario en DataFrame y lo imprimimos
        df=pd.DataFrame(list(account_info_dict.items()),columns=['property','value'])
        print("account_info() as dataframe:")
        print(df)
else:
    print("failed to connect to trade account 25115284 with password=gqz0343lbdm, erro

# finalizamos la conexión con el terminal MetaTrader 5
```

© 2000-2025, MetaQuotes Ltd.

```
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
Show account_info()._asdict():
  login=25115284
  trade_mode=0
  leverage=100
  limit_orders=200
  margin_so_mode=0
  trade_allowed=True
  trade_expert=True
  margin_mode=2
  currency_digits=2
  fifo_close=False
  balance=99511.4
  credit=0.0
  profit=41.82
  equity=99553.22
  margin=98.18
  margin_free=99455.04
  margin_level=101398.67590140559
  margin_so_call=50.0
  margin_so_so=30.0
  margin_initial=0.0
  margin_maintenance=0.0
  assets=0.0
  liabilities=0.0
  commission_blocked=0.0
  server=MetaQuotes-Demo
  currency=USD
  company=MetaQuotes Software Corp.

account_info() as dataframe
              property                     value
0                 login                  25115284
1            trade_mode                         0
2              leverage                       100
3          limit_orders                       200
4       margin_so_mode                         0
5         trade_allowed                      True
6          trade_expert                      True
7           margin_mode                         2
8      currency_digits                         2
9            fifo_close                     False
10             balance                   99588.3
```

© 2000-2025, MetaQuotes Ltd.

```
11              credit                          0
12              profit                     -45.13
13              equity                    99543.2
14              margin                      54.37
15         margin_free                    99488.8
16        margin_level                     183085
17      margin_so_call                         50
18        margin_so_so                         30
19      margin_initial                          0
20  margin_maintenance                          0
21              assets                          0
22         liabilities                          0
23  commission_blocked                          0
24                name                 James Smith
25              server              MetaQuotes-Demo
26            currency                         USD
27             company  MetaQuotes Software Corp.
```

Ver también

initialize, shutdown, login

© 2000-2025, MetaQuotes Ltd.

terminal_info

Obtiene el estado y los ajustes de un terminal de cliente MetaTrader 5 activado.

```
terminal_info()
```

Valor retornado

Retorna la información como estructura de tuplas nombradas (namedtuple). En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función retorna en una llamada toda la información que se puede obtener con la ayuda de TerminalInfoInteger, TerminalInfoDouble y TerminalInfoDouble.

Ejemplo:

```
import MetaTrader5 as mt5
import pandas as pd
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# mostramos la información sobre la versión de MetaTrader 5
print(mt5.version())
# mostramos la información sobre los ajustes y el estado del terminal
terminal_info=mt5.terminal_info()
if terminal_info!=None:
    # mostramos como son los datos sobre el terminal
    print(terminal_info)
    # mostramos los datos en forma de lista
    print("Show terminal_info()._asdict():")
    terminal_info_dict = mt5.terminal_info()._asdict()
    for prop in terminal_info_dict:
        print("  {}={}".format(prop, terminal_info_dict[prop]))
    print()
   # transformamos el diccionario en DataFrame y lo imprimimos
    df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
    print("terminal_info() as dataframe:")
    print(df)

# finalizamos la conexión con el terminal MetaTrader 5
```

© 2000-2025, MetaQuotes Ltd.

```
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
[500, 2366, '20 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True,....
Show terminal_info()._asdict():
  community_account=True
  community_connection=True
  connected=True
  dlls_allowed=False
  trade_allowed=False
  tradeapi_disabled=False
  email_enabled=False
  ftp_enabled=False
  notifications_enabled=False
  mqid=False
  build=2366
  maxbars=5000
  codepage=1251
  ping_last=77850
  community_balance=707.10668201585
  retransmission=0.0
  company=MetaQuotes Software Corp.
  name=MetaTrader 5
  language=Russian
  path=E:\ProgramFiles\MetaTrader 5
  data_path=E:\ProgramFiles\MetaTrader 5
  commondata_path=C:\Users\Rosh\AppData\Roaming\MetaQuotes\Terminal\Common

terminal_info() as dataframe:
                 property                      value
0       community_account                       True
1    community_connection                       True
2               connected                       True
3            dlls_allowed                      False
4           trade_allowed                      False
5       tradeapi_disabled                      False
6           email_enabled                      False
7             ftp_enabled                      False
8   notifications_enabled                      False
9                    mqid                      False
10                  build                       2367
11                maxbars                       5000
12               codepage                       1251
13             ping_last                      80953
14      community_balance                    707.107
```

© 2000-2025, MetaQuotes Ltd.

```
15         retransmission                   0.063593
16                company  MetaQuotes Software Corp.
17                   name                MetaTrader 5
18               language                     Russian
```

Ver también

initialize, shutdown, version

© 2000-2025, MetaQuotes Ltd.

symbols_total

Obtiene el número total de instrumentos financieros en el terminal MetaTrader 5.

```
symbols_total()
```

Valor retornado

Valor de tipo entero.

Observación

La función es un análogo de SymbolsTotal(), pero retorna el número total de símbolos, incluyendo los de usuario y los desactivados en MarketWatch.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtenemos el número de instrumentos financieros
symbols=mt5.symbols_total()
if symbols>0:
    print("Total symbols =",symbols)
else:
    print("Symbols not found")

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
```

Ver también

symbols_get, symbol_select, symbol_info

© 2000-2025, MetaQuotes Ltd.

symbols_get

Obtiene todos los instrumentos financieros del terminal MetaTrader 5.

```
symbols_get(
   group="GROUP"      // filtro para la selección de símbolos
)
```

group="GROUP"

[in] Filtro para seleccionar un grupo solo con los símbolos necesarios. Parámetro no obligatorio. Si el grupo ha sido establecido, la función retornará solo los símbolos que cumplan con el criterio establecido.

Valor retornado

Retorna los símbolos en forma de tupla (tuple). En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

El parámetro *group* permite filtrar los símbolos por nombre. Está permitido usar '*' al inicio y el final de la línea.

El parámetro *group* se puede usar como parámetro nombrado, y también como parámetro no nombrado. Ambas variantes funcionan igual. El uso de la variante nombrada (group="GROUP") hace el código más comprensible.

El parámetro *group* puede contener varias condiciones separadas por comas. Las condiciones se pueden establecer como máscara con el uso de '*'. Para realizar exclusiones, se puede usar el símbolo de negación lógica '!'. En este caso, además, todas las condiciones se aplican de forma secuencial, es decir, primero se deben indicar las inclusiones en el grupo, y después las condiciones de exclusión. Por ejemplo, *group*="*, !EUR" significa que primero debemos seleccionar todos los símbolos y después excluir aquellos de ellos que contengan en el nombre "EUR".

Y diferencia de symbol_info(), la función symbols_get() retorna en una llamada la información de todos los símbolos solicitados.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtenemos todos los símbolos
symbols=mt5.symbols_get()
print('Symbols: ', len(symbols))
count=0
```

© 2000-2025, MetaQuotes Ltd.

```
# mostramos los 5 primeros
for s in symbols:
    count+=1
    print("{}. {}".format(count,s.name))
    if count==5: break
print()

# obtenemos los símbolos cuyos nombres contienen U
ru_symbols=mt5.symbols_get("*RU*")
print('len(*RU*): ', len(ru_symbols))
for s in ru_symbols:
    print(s.name)
print()

# obtenemos los símbolos cuyos nombres no contienen USD,EUR,JPY y GBP
group_symbols=mt5.symbols_get(group="*,!*USD*,!*EUR*,!*JPY*,!*GBP*")
print('len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):', len(group_symbols))
for s in group_symbols:
    print(s.name,":",s)

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
Symbols:  84
1. EURUSD
2. GBPUSD
3. USDCHF
4. USDJPY
5. USDCNH

len(*RU*):  8
EURUSD
USDRUB
USDRUR
EURRUR
EURRUB
FORTS.RUB.M5
EURUSD_T20
EURUSD4

len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):  13
AUDCAD : SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_dea
AUDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
AUDNZD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
CADCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
NZDCAD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
```

© 2000-2025, MetaQuotes Ltd.

```
NZDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDSGD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
CADMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
CHFMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
FORTS.RTS.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
FORTS.RUB.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
FOREX.CHF.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
```

Ver también

symbols_total, symbol_select, symbol_info

© 2000-2025, MetaQuotes Ltd.

symbol_info

Obtiene la información sobre el instrumento financiero indicado.

```
symbol_info(
   symbol      // Nombre del instrumento financiero.
)
```

symbol

[in] Nombre del instrumento financiero. Parámetro no nombrado obligatorio.

Valor retornado

Retorna la información como estructura de tuplas nombradas (namedtuple). En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función retorna en una sola llamada toda la información que se puede obtener con la ayuda de SymbolInfoInteger, SymbolInfoDouble y SymbolInfoString.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# intentamos activar la muestra del símbolo EURJPY en MarketWatch
selected=mt5.symbol_select("EURJPY",True)
if not selected:
    print("Failed to select EURJPY")
    mt5.shutdown()
    quit()

# mostramos las propiedades del símbolo EURJPY
symbol_info=mt5.symbol_info("EURJPY")
if symbol_info!=None:
    # mostramos como son los datos sobre el terminal
    print(symbol_info)
    print("EURJPY: spread =",symbol_info.spread,"  digits =",symbol_info.digits)
    # mostramos las propiedades del símbolo en forma de lista
    print("Show symbol_info(\"EURJPY\")._asdict():")
    symbol_info_dict = mt5.symbol_info("EURJPY")._asdict()
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
```

© 2000-2025, MetaQuotes Ltd.

```
# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0, ses
EURJPY: spread = 17   digits = 3
Show symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585069682
  digits=3
  spread=17
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
  option_mode=0
  option_right=0
  bid=120.024
  bidhigh=120.506
  bidlow=118.798
  ask=120.041
  askhigh=120.526
  asklow=118.828
  last=0.0
```

© 2000-2025, MetaQuotes Ltd.

```
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=0.001
trade_tick_value=0.8977708350166538
trade_tick_value_profit=0.8977708350166538
trade_tick_value_loss=0.8978272580355541
trade_tick_size=0.001
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-0.2
swap_short=-1.2
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
price_greeks_gamma=0.0
price_greeks_vega=0.0
price_greeks_rho=0.0
price_greeks_omega=0.0
price_sensitivity=0.0
basis=
category=
currency_base=EUR
currency_profit=JPY
```

© 2000-2025, MetaQuotes Ltd.

```
currency_margin=EUR
bank=
description=Euro vs Japanese Yen
exchange=
formula=
isin=
name=EURJPY
page=http://www.google.com/finance?q=EURJPY
path=Forex\EURJPY
```

Ver también

account_info, terminal_info

© 2000-2025, MetaQuotes Ltd.

symbol_info_tick

Obtiene el último tick del instrumento financiero indicado.

```
symbol_info_tick(
   symbol      // Nombre del instrumento financiero.
)
```

symbol

[in] Nombre del instrumento financiero. Parámetro no nombrado obligatorio.

Valor retornado

Retorna la información en forma de tupla (tuple). En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función es un análogo de SymbolInfoTick.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# intentamos activar la muestra del símbolo GBPUSD en MarketWatch
selected=mt5.symbol_select("GBPUSD",True)
if not selected:
    print("Failed to select GBPUSD")
    mt5.shutdown()
    quit()

# mostramos el último tick del símbolo GBPUSD
lasttick=mt5.symbol_info_tick("GBPUSD")
print(lasttick)
# mostramos los valores de los campos del tick en forma de lista
print("Show symbol_info_tick(\"GBPUSD\")._asdict():")
symbol_info_tick_dict = mt5.symbol_info_tick("GBPUSD")._asdict()
for prop in symbol_info_tick_dict:
    print("  {}={}".format(prop, symbol_info_tick_dict[prop]))

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

```
Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
Tick(time=1585070338, bid=1.17264, ask=1.17279, last=0.0, volume=0, time_msc=158507033
Show symbol_info_tick._asdict():
  time=1585070338
  bid=1.17264
  ask=1.17279
  last=0.0
  volume=0
  time_msc=1585070338728
  flags=2
  volume_real=0.0
```

Ver también

symbol_info, symbol_info

© 2000-2025, MetaQuotes Ltd.

symbol_select

Selecciona un símbolo en la ventana MarketWatch o quita un símbolo de esta ventana.

```
symbol_select(
   symbol,       // nombre del instrumento financiero
   enable=None   // activar o desactivar
)
```

symbol

[in] Nombre del instrumento financiero. Parámetro no nombrado obligatorio.

enable

[in] Conmutador. Parámetro no nombrado no obligatorio. Si el valor es false, el símbolo deberá ser eliminado de la ventana de MarketWatch, en caso contrario, el símbolo deberá ser seleccionado en la ventana de MarketWatch. El símbolo no podrá ser eliminado si hay gráficos abiertos con este símbolo, o si hay posiciones abiertas de este símbolo.

Valor retornado

True si se ha ejecutado con éxito, de lo contrario, False.

Observación

La función es un análogo de SymbolSelect.

Ejemplo:

```
import MetaTrader5 as mt5
import pandas as pd
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# intentamos activar la muestra del símbolo EURCAD en MarketWatch
selected=mt5.symbol_select("EURCAD",True)
if not selected:
    print("Failed to select EURCAD, error code =",mt5.last_error())
else:
    symbol_info=mt5.symbol_info("EURCAD")
    print(symbol_info)
    print("EURCAD: currency_base =",symbol_info.currency_base,"  currency_profit =",sy
    print()

    # obtenemos las propiedades del símbolo en forma de diccionario
    print("Show symbol_info()._asdict():")
    symbol_info_dict = symbol_info._asdict()
```

© 2000-2025, MetaQuotes Ltd.

```
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
    print()

   # transformamos el diccionario en DataFrame y lo imprimimos
    df=pd.DataFrame(list(symbol_info_dict.items()),columns=['property','value'])
    print("symbol_info_dict() as dataframe:")
    print(df)

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0, ses
EURCAD: currency_base = EUR   currency_profit = CAD   currency_margin = EUR

Show symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585217595
  digits=5
  spread=39
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
```

© 2000-2025, MetaQuotes Ltd.

```
option_mode=0
option_right=0
bid=1.55192
bidhigh=1.55842
bidlow=1.5419800000000001
ask=1.5523099999999999
askhigh=1.55915
asklow=1.5436299999999998
last=0.0
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=1e-05
trade_tick_value=0.7043642408362214
trade_tick_value_profit=0.7043642408362214
trade_tick_value_loss=0.7044535553770941
trade_tick_size=1e-05
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-1.1
swap_short=-0.9
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
```

© 2000-2025, MetaQuotes Ltd.

```
  price_greeks_gamma=0.0
  price_greeks_vega=0.0
  price_greeks_rho=0.0
  price_greeks_omega=0.0
  price_sensitivity=0.0
  basis=
  category=
  currency_base=EUR
  currency_profit=CAD
  currency_margin=EUR
  bank=
  description=Euro vs Canadian Dollar
  exchange=
  formula=
  isin=
  name=EURCAD
  page=http://www.google.com/finance?q=EURCAD
  path=Forex\EURCAD

symbol_info_dict() as dataframe:
         property                                      value
0          custom                                      False
1      chart_mode                                          0
2          select                                       True
3         visible                                       True
4   session_deals                                          0
..            ...                                        ...
91        formula
92           isin
93           name                                     EURCAD
94           page  http://www.google.com/finance?q=EURCAD
95           path                               Forex\EURCAD

[96 rows x 2 columns]
```

Ver también

symbol_info

© 2000-2025, MetaQuotes Ltd.

market_book_add

Realiza la suscripción del terminal MetaTrader 5 para recibir eventos sobre los cambios en la profundidad de mercado del símbolo indicado.

```
market_book_add(
   symbol      // nombre del instrumento financiero
)
```

symbol

[in] Nombre del instrumento financiero. Parámetro no nombrado obligatorio.

Valor retornado

True si se ha ejecutado con éxito, de lo contrario, False.

Observación

La función es un análogo de MarketBookAdd.

Ver también

market_book_get, market_book_release, Estructura de la profundidad de mercado

© 2000-2025, MetaQuotes Ltd.

market_book_get

Retorna desde BookInfo la tupla que contiene las entradas de la profundidad de mercado del símbolo indicado.

```
market_book_get(
   symbol      // nombre del instrumento financiero
)
```

symbol

[in] Nombre del instrumento financiero. Parámetro no nombrado obligatorio.

Valor retornado

Retorna el contenido de la profundidad de mercado en forma de tupla de entradas BookInfo que contienen el tipo de solicitud, el precio y el volumen en lotes. BookInfo es un análogo de la estructura MqlBookInfo.

En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

Previamente, deberemos usar la función market_book_add(), encargada de realizar la suscripción para obtener los eventos de cambio en la profundidad de mercado.

La función es un análogo de MarketBookGet.

Ejemplo:

```
import MetaTrader5 as mt5
import time
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print("")

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
   # finalizamos la conexión con el terminal MetaTrader 5
    mt5.shutdown()
    quit()

# nos suscribimos a la obtención de actualizaciones en la profundidad de mercado del s
if mt5.market_book_add('EURUSD'):
  # obtenemos 10 veces en un ciclo los datos de la profundidad de mercado
   for i in range(10):
        # obtenemos el contenido de la profundidad de mercado (Depth of Market)
        items = mt5.market_book_get('EURUSD')
        # mostramos la profundidad de mercado completa con una sola línea como está
        print(items)
```

© 2000-2025, MetaQuotes Ltd.

```
        # ahora, mostramos cada solicitud aparte, para mayor visibilidad
        if items:
            for it in items:
                # contenido de la solicitud
                print(it._asdict())
        # hacemos una pausa de 5 segundos antes de la siguiente solicitud de la profur
        time.sleep(5)
  # cancelamos la suscripción a las actualizaciones en la profundidad de mercado (Dept
   mt5.market_book_release('EURUSD')
else:
    print("mt5.market_book_add('EURUSD') failed, error code =",mt5.last_error())

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.34

(BookInfo(type=1, price=1.20038, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20038, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20032, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20023, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20017, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0), BookInfo(ty
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0), BookInfo(ty
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20036, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20036, 'volume': 250, 'volume_dbl': 250.0}
```

© 2000-2025, MetaQuotes Ltd.

```
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20028, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20035, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20035, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20027, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20037, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20037, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20031, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20022, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20016, 'volume': 250, 'volume_dbl': 250.0}
```

Ver también

market_book_add, market_book_release, Estructura de la profundidad de mercado

© 2000-2025, MetaQuotes Ltd.

market_book_release

Cancela la suscripción del terminal MetaTrader 5 para recibir eventos sobre los cambios en la profundidad de mercado del símbolo indicado.

```
market_book_release(
   symbol      // nombre del instrumento financiero
)
```

symbol

[in] Nombre del instrumento financiero. Parámetro no nombrado obligatorio.

Valor retornado

True si se ha ejecutado con éxito, de lo contrario, False.

Observación

La función es un análogo de MarketBookRelease.

Ver también

market_book_add, market_book_get, Estructura de la profundidad de mercado

© 2000-2025, MetaQuotes Ltd.

copy_rates_from

Obtiene las barras del terminal MetaTrader 5, a partir de la fecha indicada.

```
copy_rates_from(
   symbol,       // nombre del símbolo
   timeframe,    // marco temporal
   date_from,    // fecha de apertura de la barra inicial
   count         // número de barras
   )
```

Parámetros

symbol

[in] Nombre del instrumento financiero, por ejemplo, "EURUSD". Parámetro no nombrado obligatorio.

timeframe

[in] Marco temporal para el que se solicitan las barras. Se establece con el valor de la enumeración TIMEFRAME. Parámetro no nombrado obligatorio.

date_from

[in] Fecha de apertura de la primera barra de la muestra solicitada. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio.

count

[in] Número de barras que se deben obtener. Parámetro no nombrado obligatorio.

Valor retornado

Retona las barras en forma de matriz numpy con las columnas nombradas time, open, high, low, close, tick_volume, spread y real_volume. En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

Para más información, mire la función CopyRates().

Sólo se devuelven los datos cuya fecha es menor (anterior) o igual a la especificada. Este intervalo se establece y se cuenta con la precisión de segundos. Es decir, la fecha de apertura de cualquier barra siempre es igual o menor que la especificada.

El terminal MetaTrader 5 entrega las barras solo dentro de la historia disponible para el usuario en los gráficos. El número de barras disponibles para el usuario se establece en los ajustes con el parámetro "Máx. de barras en la ventana".

Python, al crear el objeto datetime, usa el huso horario de la hora local, mientras que el terminal MetaTrader 5 guarda la hora de los ticks y la apertura de las barras en el huso horario UTC (sin desplazamiento). Por eso, para ejecutar las funciones que usen la hora, es necesario crear datetime en la hora UTC. Los datos recibidos del terminal MetaTrader 5 tienen la hora UTC.

TIMEFRAME es una enumeración con los posibles valores de los periodos del gráfico

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
TIMEFRAME_M1	1 minuto
TIMEFRAME_M2	2 minutos
TIMEFRAME_M3	3 minutos
TIMEFRAME_M4	4 minutos
TIMEFRAME_M5	5 minutos
TIMEFRAME_M6	6 minutos
TIMEFRAME_M10	10 minutos
TIMEFRAME_M12	12 minutos
TIMEFRAME_M12	15 minutos
TIMEFRAME_M20	20 minutos
TIMEFRAME_M30	30 minutos
TIMEFRAME_H1	1 hora
TIMEFRAME_H2	2 horas
TIMEFRAME_H3	3 horas
TIMEFRAME_H4	4 horas
TIMEFRAME_H6	6 horas
TIMEFRAME_H8	8 horas
TIMEFRAME_H12	12 horas
TIMEFRAME_D1	1 día
TIMEFRAME_W1	1 semana
TIMEFRAME_MN1	1 mes

Ejemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importamos el módulo pandas para mostrar los datos obtenidos en forma de recuadro
import pandas as pd
pd.set_option('display.max_columns', 500) # cuántas columnas mostramos
pd.set_option('display.width', 1500)      # anchura máx. del recuadro para la muestra
# importamos el módulo pytz para trabajar con el huso horario
```

© 2000-2025, MetaQuotes Ltd.

```
import pytz

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# establecemos el huso horario en UTC
timezone = pytz.timezone("Etc/UTC")
# creamos el objeto datetime en el huso horario UTC, para que no se aplique el desplaz
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# obtenemos 10 barras de EURUSD H4 a partir del 01.10.2020 en el huso horario UTC
rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_H4, utc_from, 10)

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
# mostramos cada elemento de los datos obtenidos en una nueva línea
print("Mostramos los datos obtenidos como son")
for rate in rates:
    print(rate)

# creamos un DataFrame de los datos obtenidos
rates_frame = pd.DataFrame(rates)
# convertimos la hora en segundos al formato datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# mostramos los datos
print("\nMostramos el frame de datos con la información")
print(rates_frame)
```

Resultado:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Mostramos los datos obtenidos como son
(1578484800, 1.11382, 1.11385, 1.1111, 1.11199, 9354, 1, 0)
(1578499200, 1.11199, 1.11308, 1.11086, 1.11179, 10641, 1, 0)
(1578513600, 1.11178, 1.11178, 1.11016, 1.11053, 4806, 1, 0)
(1578528000, 1.11053, 1.11193, 1.11033, 1.11173, 3480, 1, 0)
(1578542400, 1.11173, 1.11189, 1.11126, 1.11182, 2236, 1, 0)
(1578556800, 1.11181, 1.11203, 1.10983, 1.10993, 7984, 1, 0)
(1578571200, 1.10994, 1.11173, 1.10965, 1.11148, 7406, 1, 0)
(1578585600, 1.11149, 1.11149, 1.10923, 1.11046, 7468, 1, 0)
(1578600000, 1.11046, 1.11097, 1.11033, 1.11051, 3450, 1, 0)
(1578614400, 1.11051, 1.11093, 1.11017, 1.11041, 2448, 1, 0)

Mostramos el frame de datos con la información
                 time     open     high      low    close  tick_volume  spread  real_v
0 2020-01-08 12:00:00  1.11382  1.11385  1.11110  1.11199         9354       1
```

© 2000-2025, MetaQuotes Ltd.

```
1 2020-01-08 16:00:00  1.11199  1.11308  1.11086  1.11179        10641       1
2 2020-01-08 20:00:00  1.11178  1.11178  1.11016  1.11053         4806       1
3 2020-01-09 00:00:00  1.11053  1.11193  1.11033  1.11173         3480       1
4 2020-01-09 04:00:00  1.11173  1.11189  1.11126  1.11182         2236       1
5 2020-01-09 08:00:00  1.11181  1.11203  1.10983  1.10993         7984       1
6 2020-01-09 12:00:00  1.10994  1.11173  1.10965  1.11148         7406       1
7 2020-01-09 16:00:00  1.11149  1.11149  1.10923  1.11046         7468       1
8 2020-01-09 20:00:00  1.11046  1.11097  1.11033  1.11051         3450       1
9 2020-01-10 00:00:00  1.11051  1.11093  1.11017  1.11041         2448       1
```

Ver también

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_rates_from_pos

Obtiene las barras del terminal MetaTrader 5, a partir del índice establecido.

```
copy_rates_from_pos(
   symbol,       // nombre del símbolo
   timeframe,    // marco temporal
   start_pos,    // número de la barra inicial
   count         // número de barras
   )
```

Parámetros

symbol

[in] Nombre del instrumento financiero, por ejemplo, "EURUSD". Parámetro no nombrado obligatorio.

timeframe

[in] Marco temporal para el que se solicitan las barras. Se establece con el valor de la enumeración TIMEFRAME. Parámetro no nombrado obligatorio.

start_pos

[in] Número inicial de la barra, a partir del cual se solicitan los datos. La numeración de las barras va del presente hacia el pasado, es decir, la barra cero es la actual. Parámetro no nombrado obligatorio.

count

[in] Número de barras que se deben obtener. Parámetro no nombrado obligatorio.

Valor retornado

Retona las barras en forma de matriz numpy con las columnas nombradas time, open, high, low, close, tick_volume, spread y real_volume. En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

Para más información, mire la función CopyRates().

El terminal MetaTrader 5 entrega las barras solo dentro de la historia disponible para el usuario en los gráficos. El número de barras disponibles para el usuario se establece en los ajustes con el parámetro "Máx. de barras en la ventana".

Ejemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importamos el módulo pandas para mostrar los datos obtenidos en forma de recuadro
import pandas as pd
```

© 2000-2025, MetaQuotes Ltd.

```
pd.set_option('display.max_columns', 500) # cuántas columnas mostramos
pd.set_option('display.width', 1500)      # anchura máx. del recuadro para la muestra

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# solicitamos 10 barras de GBPUSD D1 desde el día actual
rates = mt5.copy_rates_from_pos("GBPUSD", mt5.TIMEFRAME_D1, 0, 10)

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()

# mostramos cada elemento de los datos obtenidos en una nueva línea
print("Mostramos los datos obtenidos como son")
for rate in rates:
    print(rate)

# creamos un DataFrame de los datos obtenidos
rates_frame = pd.DataFrame(rates)
# convertimos la hora en segundos al formato datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# mostramos los datos
print("\nMostramos el frame de datos con la información")
print(rates_frame)
```

Resultado:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Mostramos los datos obtenidos como son
(1581552000, 1.29568, 1.30692, 1.29441, 1.30412, 68228, 0, 0)
(1581638400, 1.30385, 1.30631, 1.3001, 1.30471, 56498, 0, 0)
(1581897600, 1.30324, 1.30536, 1.29975, 1.30039, 49400, 0, 0)
(1581984000, 1.30039, 1.30486, 1.29705, 1.29952, 62288, 0, 0)
(1582070400, 1.29952, 1.3023, 1.29075, 1.29187, 57909, 0, 0)
(1582156800, 1.29186, 1.29281, 1.28489, 1.28792, 61033, 0, 0)
(1582243200, 1.28802, 1.29805, 1.28746, 1.29566, 66386, 0, 0)
(1582502400, 1.29426, 1.29547, 1.28865, 1.29283, 66933, 0, 0)
(1582588800, 1.2929, 1.30178, 1.29142, 1.30037, 80121, 0, 0)
(1582675200, 1.30036, 1.30078, 1.29136, 1.29374, 49286, 0, 0)

Mostramos el frame de datos con la información
        time     open     high      low    close  tick_volume  spread  real_volume
0 2020-02-13  1.29568  1.30692  1.29441  1.30412        68228       0            0
1 2020-02-14  1.30385  1.30631  1.30010  1.30471        56498       0            0
2 2020-02-17  1.30324  1.30536  1.29975  1.30039        49400       0            0
```

© 2000-2025, MetaQuotes Ltd.

3	2020-02-18	1.30039	1.30486	1.29705	1.29952	62288	0	0
4	2020-02-19	1.29952	1.30230	1.29075	1.29187	57909	0	0
5	2020-02-20	1.29186	1.29281	1.28489	1.28792	61033	0	0
6	2020-02-21	1.28802	1.29805	1.28746	1.29566	66386	0	0
7	2020-02-24	1.29426	1.29547	1.28865	1.29283	66933	0	0
8	2020-02-25	1.29290	1.30178	1.29142	1.30037	80121	0	0
9	2020-02-26	1.30036	1.30078	1.29136	1.29374	49286	0	0

Ver también

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_rates_range

Obtiene las barras en el intervalo de fechas indicado del terminal MetaTrader 5.

```
copy_rates_range(
   symbol,       // nombre del símbolo
   timeframe,    // marco temporal
   date_from,    // fecha a partir de la cual se solicitan las barras
   date_to       // fecha hasta la cual se solicitan las barras
   )
```

Parámetros

symbol

[in] Nombre del instrumento financiero, por ejemplo, "EURUSD". Parámetro no nombrado obligatorio.

timeframe

[in] Marco temporal para el que se solicitan las barras. Se establece con el valor de la enumeración TIMEFRAME. Parámetro no nombrado obligatorio.

date_from

[in] Fecha a partir de la cual se solicitan las barras. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Se dan las barras con una hora de apertura >= date_from. Parámetro no nombrado obligatorio.

date_to

[in] Fecha hasta la cual se solicitan las barras. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Se dan las barras con una hora de apertura <= date_to. Parámetro no nombrado obligatorio.

Valor retornado

Retona las barras en forma de matriz numpy con las columnas nombradas time, open, high, low, close, tick_volume, spread y real_volume. En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

Para más información, mire la función CopyRates().

El terminal MetaTrader 5 entrega las barras solo dentro de la historia disponible para el usuario en los gráficos. El número de barras disponibles para el usuario se establece en los ajustes con el parámetro "Máx. de barras en la ventana".

Python, al crear el objeto datetime, usa el huso horario de la hora local, mientras que el terminal MetaTrader 5 guarda la hora de los ticks y la apertura de las barras en el huso horario UTC (sin desplazamiento). Por eso, para ejecutar las funciones que usen la hora, es necesario crear datetime en la hora UTC. Los datos recibidos del terminal MetaTrader 5 tienen la hora UTC.

Ejemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
```

© 2000-2025, MetaQuotes Ltd.

```
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importamos el módulo pandas para mostrar los datos obtenidos en forma de recuadro
import pandas as pd
pd.set_option('display.max_columns', 500) # cuántas columnas mostramos
pd.set_option('display.width', 1500)      # máx. anchura del recuadro para la muestra
# importamos el módulo pytz para trabajar con el huso horario
import pytz

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# establecemos el huso horario en UTC
timezone = pytz.timezone("Etc/UTC")
# creamos los objetos datetime en el huso horario UTC, para que no se aplique el despl
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, hour = 13, tzinfo=timezone)
# obtenemos las barras de USDJPY M5 en el intervalo 2020.01.10 00:00 - 2020.01.11 13:(
rates = mt5.copy_rates_range("USDJPY", mt5.TIMEFRAME_M5, utc_from, utc_to)

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
# mostramos cada elemento de los datos obtenidos en una nueva línea
print("Mostramos los datos obtenidos como son")
counter=0
for rate in rates:
    counter+=1
    if counter<=10:
        print(rate)

# creamos un DataFrame de los datos obtenidos
rates_frame = pd.DataFrame(rates)
# convertimos la hora en segundos al formato datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# mostramos los datos
print("\nMostramos el frame de datos con la información")
print(rates_frame.head(10))
```

Resultado:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Mostramos los datos obtenidos como son
(1578614400, 109.513, 109.527, 109.505, 109.521, 43, 2, 0)
(1578614700, 109.521, 109.549, 109.518, 109.543, 215, 8, 0)
(1578615000, 109.543, 109.543, 109.466, 109.505, 98, 10, 0)
(1578615300, 109.504, 109.534, 109.502, 109.517, 155, 8, 0)
(1578615600, 109.517, 109.539, 109.513, 109.527, 71, 4, 0)
(1578615900, 109.526, 109.537, 109.484, 109.52, 106, 9, 0)
(1578616200, 109.52, 109.524, 109.508, 109.51, 205, 7, 0)
```

© 2000-2025, MetaQuotes Ltd.

```
(1578616500, 109.51, 109.51, 109.491, 109.496, 44, 8, 0)
(1578616800, 109.496, 109.509, 109.487, 109.5, 85, 5, 0)
(1578617100, 109.5, 109.504, 109.487, 109.489, 82, 7, 0)

Mostramos el frame de datos con la información
                 time     open     high      low    close  tick_volume  spread  real_v
0 2020-01-10 00:00:00  109.513  109.527  109.505  109.521           43       2
1 2020-01-10 00:05:00  109.521  109.549  109.518  109.543          215       8
2 2020-01-10 00:10:00  109.543  109.543  109.466  109.505           98      10
3 2020-01-10 00:15:00  109.504  109.534  109.502  109.517          155       8
4 2020-01-10 00:20:00  109.517  109.539  109.513  109.527           71       4
5 2020-01-10 00:25:00  109.526  109.537  109.484  109.520          106       9
6 2020-01-10 00:30:00  109.520  109.524  109.508  109.510          205       7
7 2020-01-10 00:35:00  109.510  109.510  109.491  109.496           44       8
8 2020-01-10 00:40:00  109.496  109.509  109.487  109.500           85       5
9 2020-01-10 00:45:00  109.500  109.504  109.487  109.489           82       7
```

Ver también

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_ticks_from

Obtiene los ticks del terminal MetaTrader 5, a partir de la fecha indicada.

```
copy_ticks_from(
   symbol,       // nombre del símbolo
   date_from,    // fecha a partir de la cual se solicitan los ticks
   count,        // número de ticks solicitados
   flags         // combinación de banderas que determina el tipo de ticks solicitados
   )
```

Parámetros

symbol

[in] Nombre del instrumento financiero, por ejemplo, "EURUSD". Parámetro no nombrado obligatorio.

date_from

[in] Fecha a partir de la cual se solicitan los ticks. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio.

count

[in] Número de ticks que se deben obtener. Parámetro no nombrado obligatorio.

flags

[in] Bandera que determina el tipo de ticks solicitados. COPY_TICKS_INFO - ticks provocados por los cambios de Bid y/o Ask, COPY_TICKS_TRADE - ticks con los cambios de Last y Volume, COPY_TICKS_ALL - todos los ticks. Los valores de las banderas se describen en la enumeración COPY_TICKS. Parámetro no nombrado obligatorio.

Valor retornado

Retona los ticks en forma de matriz numpy con las columnas nombradas time, bid, ask, last y flags. El valor flags puede ser una combinación de banderas de la enumeración TICK_FLAG. En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

Para más información, mire la función CopyTicks.

Python, al crear el objeto datetime, usa el huso horario de la hora local, mientras que el terminal MetaTrader 5 guarda la hora de los ticks y la apertura de las barras en el huso horario UTC (sin desplazamiento). Por eso, para ejecutar las funciones que usen la hora, es necesario crear datetime en la hora UTC. Los datos recibidos del terminal MetaTrader 5 tienen la hora UTC.

COPY_TICKS determina los tipos de ticks que pueden ser solicitados con la ayuda de las funciones copy_ticks_from() y copy_ticks_range().

Identificador	Descripción
COPY_TICKS_ALL	todos los ticks

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
COPY_TICKS_INFO	ticks que contengan cambios en los precios Bid y/o Ask
COPY_TICKS_TRADE	ticks que contengan cambios en el precio Last y/o el volumen (Volume)

TICK_FLAG determina las posibles banderas para los ticks. Estas banderas se usan para describir los ticks obtenidos con las funciones copy_ticks_from() y copy_ticks_range().

Identificador	Descripción
TICK_FLAG_BID	el precio Bid ha cambiado
TICK_FLAG_ASK	el precio Ask ha cambiado
TICK_FLAG_LAST	el precio Last ha cambiado
TICK_FLAG_VOLUME	el volumen (Volume) ha cambiado
TICK_FLAG_BUY	el precio de la última compra (Buy) ha cambiado
TICK_FLAG_SELL	el precio de la última venta (Sell) ha cambiado

Ejemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importamos el módulo pandas para mostrar los datos obtenidos en forma de recuadro
import pandas as pd
pd.set_option('display.max_columns', 500) # cuántas columnas mostramos
pd.set_option('display.width', 1500)      # anchura máx. del recuadro para la muestra
# importamos el módulo pytz para trabajar con el huso horario
import pytz

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2025, MetaQuotes Ltd.

```
    quit()

# establecemos el huso horario en UTC
timezone = pytz.timezone("Etc/UTC")
# creamos el objeto datetime en el huso horario UTC, para que no se aplique el desplaz
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# solicitamos 100000 ticks de EURUSD desde 10.01.2019 en el huso horario UTC
ticks = mt5.copy_ticks_from("EURUSD", utc_from, 100000, mt5.COPY_TICKS_ALL)
print("Ticks obtenidos:",len(ticks))

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()

# mostramos los datos de cada tick en una nueva línea
print("Mostramos los datos obtenidos como son")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# creamos un DataFrame de los datos obtenidos
ticks_frame = pd.DataFrame(ticks)
# convertimos la hora en segundos al formato datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# mostramos los datos
print("\nMostramos el frame de datos con los ticks")
print(ticks_frame.head(10))
```

```
Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Ticks obtenidos: 100000
Mostramos los datos obtenidos como son
(1578614400, 1.11051, 1.11069, 0., 0, 1578614400987, 134, 0.)
(1578614402, 1.11049, 1.11067, 0., 0, 1578614402025, 134, 0.)
(1578614404, 1.1105, 1.11066, 0., 0, 1578614404057, 134, 0.)
(1578614404, 1.11049, 1.11067, 0., 0, 1578614404344, 134, 0.)
(1578614412, 1.11052, 1.11064, 0., 0, 1578614412106, 134, 0.)
(1578614418, 1.11039, 1.11051, 0., 0, 1578614418265, 134, 0.)
(1578614418, 1.1104, 1.1105, 0., 0, 1578614418905, 134, 0.)
(1578614419, 1.11039, 1.11051, 0., 0, 1578614419519, 134, 0.)
(1578614456, 1.11037, 1.11065, 0., 0, 1578614456011, 134, 0.)
(1578614456, 1.11039, 1.11051, 0., 0, 1578614456015, 134, 0.)

Mostramos el frame de datos con los ticks
                 time      bid      ask  last  volume       time_msc  flags  volume_re
0 2020-01-10 00:00:00  1.11051  1.11069   0.0       0  1578614400987    134          0
1 2020-01-10 00:00:02  1.11049  1.11067   0.0       0  1578614402025    134          0
2 2020-01-10 00:00:04  1.11050  1.11066   0.0       0  1578614404057    134          0
3 2020-01-10 00:00:04  1.11049  1.11067   0.0       0  1578614404344    134          0
```

© 2000-2025, MetaQuotes Ltd.

```
4 2020-01-10 00:00:12  1.11052  1.11064   0.0       0  1578614412106    134          0
5 2020-01-10 00:00:18  1.11039  1.11051   0.0       0  1578614418265    134          0
6 2020-01-10 00:00:18  1.11040  1.11050   0.0       0  1578614418905    134          0
7 2020-01-10 00:00:19  1.11039  1.11051   0.0       0  1578614419519    134          0
8 2020-01-10 00:00:56  1.11037  1.11065   0.0       0  1578614456011    134          0
9 2020-01-10 00:00:56  1.11039  1.11051   0.0       0  1578614456015    134          0
```

Ver también

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_ticks_range

Obtiene los ticks en el intervalo de fechas indicado del terminal MetaTrader 5.

```
copy_ticks_range(
   symbol,       // nombre del símbolo
   date_from,    // fecha a partir de la cual se solicitan los ticks
   date_to,      // fecha hasta la cual se solicitan los ticks
   flags         // combinación de banderas que determina el tipo de ticks solicitados
   )
```

Parámetros

symbol

[in] Nombre del instrumento financiero, por ejemplo, "EURUSD". Parámetro no nombrado obligatorio.

date_from

[in] Fecha a partir de la cual se solicitan los ticks. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio.

date_to

[in] Fecha hasta la cual se solicitan los ticks. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio.

flags

[in] Bandera que determina el tipo de ticks solicitados. COPY_TICKS_INFO - ticks provocados por los cambios de Bid y/o Ask, COPY_TICKS_TRADE - ticks con los cambios de Last y Volume, COPY_TICKS_ALL - todos los ticks. Los valores de las banderas se describen en la enumeración COPY_TICKS. Parámetro no nombrado obligatorio.

Valor retornado

Retona los ticks en forma de matriz numpy con las columnas nombradas time, bid, ask, last y flags. El valor flags puede ser una combinación de banderas de la enumeración TICK_FLAG. En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

Para más información, mire la función CopyTicks.

Python, al crear el objeto datetime, usa el huso horario de la hora local, mientras que el terminal MetaTrader 5 guarda la hora de los ticks y la apertura de las barras en el huso horario UTC (sin desplazamiento). Por eso, para ejecutar las funciones que usen la hora, es necesario crear datetime en la hora UTC. Los datos recibidos del terminal MetaTrader 5 tienen la hora UTC, pero al intentar imprimirlos, Python aplicará el desplazamiento a la hora local. Por eso, los datos obtenidos también deben ser corregidos para la representación visual.

Ejemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
```

© 2000-2025, MetaQuotes Ltd.

```
print("MetaTrader5 package version: ",mt5.__version__)

# importamos el módulo pandas para mostrar los datos obtenidos en forma de recuadro
import pandas as pd
pd.set_option('display.max_columns', 500) # cuántas columnas mostramos
pd.set_option('display.width', 1500)      # máx. anchura del recuadro para la muestra
# importamos el módulo pytz para trabajar con el huso horario
import pytz

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# establecemos el huso horario en UTC
timezone = pytz.timezone("Etc/UTC")
# creamos los objetos datetime en el huso horario UTC, para que no se aplique el despl
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, tzinfo=timezone)
# solicitamos los ticks de AUDUSD en el intervalo 11.01.2020 - 11.01.2020
ticks = mt5.copy_ticks_range("AUDUSD", utc_from, utc_to, mt5.COPY_TICKS_ALL)
print("Ticks obtenidos:",len(ticks))

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
# mostramos los datos de cada tick en una nueva línea
print("Mostramos los datos obtenidos como son")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# creamos un DataFrame de los datos obtenidos
ticks_frame = pd.DataFrame(ticks)
# convertimos la hora en segundos al formato datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# mostramos los datos
print("\nMostramos el frame de datos con los ticks")
print(ticks_frame.head(10))
```

Resultado:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Ticks obtenidos: 37008
Mostramos los datos obtenidos como son
(1578614400, 0.68577, 0.68594, 0., 0, 1578614400820, 134, 0.)
(1578614401, 0.68578, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614401, 0.68575, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614411, 0.68576, 0.68594, 0., 0, 1578614411388, 130, 0.)
(1578614411, 0.68575, 0.68594, 0., 0, 1578614411560, 130, 0.)
(1578614414, 0.68576, 0.68595, 0., 0, 1578614414973, 134, 0.)
(1578614430, 0.68576, 0.68594, 0., 0, 1578614430188, 4, 0.)
```

© 2000-2025, MetaQuotes Ltd.

```
(1578614450, 0.68576, 0.68595, 0., 0, 1578614450408, 4, 0.)
(1578614450, 0.68576, 0.68594, 0., 0, 1578614450519, 4, 0.)
(1578614456, 0.68575, 0.68594, 0., 0, 1578614456363, 130, 0.)

Mostramos el frame de datos con los ticks
                 time      bid      ask  last  volume       time_msc  flags  volume_re
0 2020-01-10 00:00:00  0.68577  0.68594   0.0       0  1578614400820    134          0
1 2020-01-10 00:00:01  0.68578  0.68594   0.0       0  1578614401128    130          0
2 2020-01-10 00:00:01  0.68575  0.68594   0.0       0  1578614401128    130          0
3 2020-01-10 00:00:11  0.68576  0.68594   0.0       0  1578614411388    130          0
4 2020-01-10 00:00:11  0.68575  0.68594   0.0       0  1578614411560    130          0
5 2020-01-10 00:00:14  0.68576  0.68595   0.0       0  1578614414973    134          0
6 2020-01-10 00:00:30  0.68576  0.68594   0.0       0  1578614430188      4          0
7 2020-01-10 00:00:50  0.68576  0.68595   0.0       0  1578614450408      4          0
8 2020-01-10 00:00:50  0.68576  0.68594   0.0       0  1578614450519      4          0
9 2020-01-10 00:00:56  0.68575  0.68594   0.0       0  1578614456363    130          0
```

Ver también

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

orders_total

Obtiene el número de órdenes activas.

```
orders_total()
```

Valor retornado

Valor de tipo entero.

Observación

La función es un análogo de OrdersTotal.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# comprobamos la presencia de órdenes activas
orders=mt5.orders_total()
if orders>0:
    print("Total orders=",orders)
else:
    print("Orders not found")

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
```

Ver también

orders_get, positions_total

© 2000-2025, MetaQuotes Ltd.

orders_get

Obtiene la órdenes activas con posibilidad de filtrado según un símbolo o ticket. Existen 3 variantes de llamada.

Llamada sin parámetros. Retorna las órdenes activas de todos los símbolos.

```
orders_get()
```

Llamada con indicación del símbolo del que debemos obtener las órdenes activas.

```
orders_get(
   symbol="SYMBOL"      // nombre del símbolo
)
```

Llamada con indicación del grupo de símbolos de los que debemos obtener las órdenes activas.

```
orders_get(
   group="GROUP"        // filtro de selección de órdenes según los símbolos
)
```

Llamada con indicación del ticket de la orden.

```
orders_get(
   ticket=TICKET        // ticket
)
```

symbol="SYMBOL"

[in] Nombre del símbolo. Parámetro nombrado no obligatorio. Si se ha indicado el símbolo, el parámetro *ticket* será ignorado.

group="GROUP"

[in] Filtro para seleccionar un grupo solo con los símbolos necesarios. Parámetro nombrado no obligatorio. Si el grupo ha sido establecido, la función retornará las órdenes activas que cumplan con el criterio establecido para el nombre del símbolo.

ticket=TICKET

[in] Ticket de la orden (ORDER_TICKET). Parámetro nombrado no obligatorio.

Valor retornado

Retorna la información como estructura de tuplas nombradas (namedtuple). En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función permite obtener en una sola llamada todas las órdenes activas; sería un análogo de la unión de OrdersTotal y OrderSelect.

El parámetro *group* permite filtrar las órdenes por símbolos. Está permitido usar '*' al inicio y el final de la línea.

El parámetro *group* puede contener varias condiciones separadas por comas. Las condiciones se pueden establecer como máscara con el uso de '*'. Para realizar exclusiones, se puede usar el símbolo de negación lógica '!'. En este caso, además, todas las condiciones se aplican de forma secuencial, es decir, primero se deben indicar las inclusiones en el grupo, y después las condiciones

© 2000-2025, MetaQuotes Ltd.

de exclusión. Por ejemplo, *group*="*, !EUR" significa que primero debemos seleccionar las órdenes según todos los símbolos, y después excluir aquellos de ellos que contengan en el nombre del símbolo "EUR".

Ejemplo:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # cuántas columnas mostramos
pd.set_option('display.width', 1500)      # máx. anchura del recuadro para la muestra
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# mostramos la información sobre las órdenes activas en el símbolo GBPUSD
orders=mt5.orders_get(symbol="GBPUSD")
if orders is None:
    print("No orders on GBPUSD, error code={}".format(mt5.last_error()))
else:
    print("Total orders on GBPUSD:",len(orders))
    # mostramos todas las órdenes activas
    for order in orders:
        print(order)
print()

# obtenemos la lista de órdenes en los símbolos cuyos nombres contienen "*GBP*"
gbp_orders=mt5.orders_get(group="*GBP*")
if gbp_orders is None:
    print("No orders with group=\"*GBP*\", error code={}".format(mt5.last_error()))
else:
    print("orders_get(group=\"*GBP*\")={}".format(len(gbp_orders)))
    # mostramos estas órdenes en forma de recuadro con la ayuda de pandas.DataFrame
    df=pd.DataFrame(list(gbp_orders),columns=gbp_orders[0]._asdict().keys())
    df.drop(['time_done', 'time_done_msc', 'position_id', 'position_by_id', 'reason',
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    print(df)

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Total orders on GBPUSD: 2
```

© 2000-2025, MetaQuotes Ltd.

```
TradeOrder(ticket=554733548, time_setup=1585153667, time_setup_msc=1585153667718, time
TradeOrder(ticket=554733621, time_setup=1585153671, time_setup_msc=1585153671419, time

orders_get(group="*GBP*")=4
      ticket           time_setup  time_setup_msc  time_expiration  type  type_time  ty
0  554733548 2020-03-25 16:27:47   1585153667718                0     3          0
1  554733621 2020-03-25 16:27:51   1585153671419                0     2          0
2  554746664 2020-03-25 16:38:14   1585154294401                0     3          0
3  554746710 2020-03-25 16:38:17   1585154297022                0     2          0
```

Ver también

orders_total, positions_get

© 2000-2025, MetaQuotes Ltd.

order_calc_margin

Retorna el tamaño del margen en la divisa de la cuenta para realizar la operación comercial indicada.

```
order_calc_margin(
   action,      // tipo de orden (ORDER_TYPE_BUY o ORDER_TYPE_SELL)
   symbol,      // nombre del símbolo
   volume,      // volumen
   price        // precio de apertura
   )
```

Parámetros

action

[in] Tipo de orden, puede adoptar valores de la enumeración ORDER_TYPE. Parámetro no nombrado obligatorio.

symbol

[in] Nombre del instrumento financiero. Parámetro no nombrado obligatorio.

volume

[in] Volumen de la operación comercial. Parámetro no nombrado obligatorio.

price

[in] Precio de apertura. Parámetro no nombrado obligatorio.

Valor retornado

Valor de tipo real en caso de ejecución exitosa, de lo contrario, None. La información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función permite valorar el tamaño del margen necesario para el tipo de orden indicado en la cuenta actual y con el entorno de mercado actual, sin tener en cuenta las órdenes pendientes y las posiciones abiertas actuales. Es un análogo de OrderCalcMargin.

ORDER_TYPE

Identificador	Descripción
ORDER_TYPE_BUY	Orden de mercado de compra
ORDER_TYPE_SELL	Orden de mercado de venta
ORDER_TYPE_BUY_LIMIT	Orden pendiente Buy Limit
ORDER_TYPE_SELL_LIMIT	Orden pendiente Sell Limit
ORDER_TYPE_BUY_STOP	Orden pendiente Buy Stop
ORDER_TYPE_SELL_STOP	Orden pendiente Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	Al alcanzarse el precio de la orden, se coloca una orden pendiente Buy Limit al precio StopLimit

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
ORDER_TYPE_SELL_STOP_LIMIT	Al alcanzarse el precio de la orden, se coloca una orden pendiente Sell Limit al precio StopLimit
ORDER_TYPE_CLOSE_BY	Orden de cierre de una posición con otra opuesta.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtenemos la divisa de la cuenta
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# creamos la lista de símbolos
symbols=("EURUSD","GBPUSD","USDJPY", "USDCHF","EURJPY","GBPJPY")
print("Symbols to check margin:", symbols)
action=mt5.ORDER_TYPE_BUY
lot=0.1
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"not found, skipped")
        continue
    if not symbol_info.visible:
        print(symbol, "is not visible, trying to switch on")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}}) failed, skipped",symbol)
            continue
    ask=mt5.symbol_info_tick(symbol).ask
    margin=mt5.order_calc_margin(action,symbol,lot,ask)
    if margin != None:
        print("   {} buy {} lot margin: {} {}".format(symbol,lot,margin,account_currer
    else:
        print("order_calc_margin failed: , error code =", mt5.last_error())

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD

Symbols to check margin: ('EURUSD', 'GBPUSD', 'USDJPY', 'USDCHF', 'EURJPY', 'GBPJPY')
```

© 2000-2025, MetaQuotes Ltd.

```
    EURUSD buy 0.1 lot margin: 109.91 USD
    GBPUSD buy 0.1 lot margin: 122.73 USD
    USDJPY buy 0.1 lot margin: 100.0 USD
    USDCHF buy 0.1 lot margin: 100.0 USD
    EURJPY buy 0.1 lot margin: 109.91 USD
    GBPJPY buy 0.1 lot margin: 122.73 USD
```

Ver también

order_calc_profit, order_check

© 2000-2025, MetaQuotes Ltd.

order_calc_profit

Retorna el tamaño del beneficio en la divisa de la cuenta para la operación comercial indicada.

```
order_calc_profit(
   action,          // tipo de orden (ORDER_TYPE_BUY o ORDER_TYPE_SELL)
   symbol,          // nombre del símbolo
   volume,          // volumen
   price_open,      // precio de apertura
   price_close      // precio de cierre
   );
```

Parámetros

action

[in] Tipo de orden, puede adoptar uno de los dos valores de la enumeración ORDER_TYPE: ORDER_TYPE_BUY o ORDER_TYPE_SELL. Parámetro no nombrado obligatorio.

symbol

[in] Nombre del instrumento financiero. Parámetro no nombrado obligatorio.

volume

[in] Volumen de la operación comercial. Parámetro no nombrado obligatorio.

price_open

[in] Precio de apertura. Parámetro no nombrado obligatorio.

price_close

[in] Precio de cierre. Parámetro no nombrado obligatorio.

Valor retornado

Valor de tipo real en caso de ejecución exitosa, de lo contrario, None. La información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función permite valorar el resultado de una operación comercial en la cuenta actual y con el entorno de mercado actual. Es un análogo de OrderCalcProfit.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2025, MetaQuotes Ltd.

```
    quit()

# obtenemos la divisa de la cuenta
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# creamos la lista de símbolos
symbols = ("EURUSD","GBPUSD","USDJPY")
print("Symbols to check margin:", symbols)
# evaluamos los valores de beneficio para las compras y las ventas
lot=1.0
distance=300
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"not found, skipped")
        continue
    if not symbol_info.visible:
        print(symbol, "is not visible, trying to switch on")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}}) failed, skipped",symbol)
            continue
    point=mt5.symbol_info(symbol).point
    symbol_tick=mt5.symbol_info_tick(symbol)
    ask=symbol_tick.ask
    bid=symbol_tick.bid
    buy_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_BUY,symbol,lot,ask,ask+distance*po
    if buy_profit!=None:
        print("   buy {} {} lot: profit on {} points => {} {}".format(symbol,lot,dista
    else:
        print("order_calc_profit(ORDER_TYPE_BUY) failed, error code =",mt5.last_error
    sell_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_SELL,symbol,lot,bid,bid-distance*
    if sell_profit!=None:
        print("   sell {} {} lots: profit on {} points => {} {}".format(symbol,lot,dis
    else:
        print("order_calc_profit(ORDER_TYPE_SELL) failed, error code =",mt5.last_erro
    print()

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD
Symbols to check margin: ('EURUSD', 'GBPUSD', 'USDJPY')
   buy EURUSD 1.0 lot: profit on 300 points => 300.0 USD
   sell EURUSD 1.0 lot: profit on 300 points => 300.0 USD

   buy GBPUSD 1.0 lot: profit on 300 points => 300.0 USD
   sell GBPUSD 1.0 lot: profit on 300 points => 300.0 USD
```

© 2000-2025, MetaQuotes Ltd.

```
buy USDJPY 1.0 lot: profit on 300 points => 276.54 USD
sell USDJPY 1.0 lot: profit on 300 points => 278.09 USD
```

Ver también

order_calc_margin, order_check

© 2000-2025, MetaQuotes Ltd.

order_check

Comprueba si los fondos son suficientes para realizar la operación comercial requerida. Los resultados de la comprobación se retornan en forma de estructura MqlTradeCheckResult.

```
order_check(
   request      // estructura de la solicitud
   );
```

Parámetros

request

[in] Estructura del tipo MqlTradeRequest que describe la acción comercial necesaria. Parámetro no nombrado obligatorio. Más abajo, podrá ver un ejemplo con el rellenado de una solicitud y la composición de una enumeración.

Valor retornado

El resultado de la comprobación se da en forma de estructura MqlTradeCheckResult. El campo *request* en la respuesta contiene la estructura de la solicitud comercial transmitida a order_check().

Observación

La comprobación exitosa de la solicitud **no indica que la operación comercial solicitada se vaya a ejecutar obligatoriamente con éxito**. La función order_check es un análogo de OrderCheck.

TRADE_REQUEST_ACTIONS

Identificador	Descripción
TRADE_ACTION_DEAL	Colocar una orden comercial para la ejecución inmediata de una transacción con los parámetros indicados (colocar una orden de mercado)
TRADE_ACTION_PENDING	Colocar una orden comercial para la ejecución de una transacción al darse las condiciones indicadas (orden pendiente)
TRADE_ACTION_SLTP	Cambiar los valores de Stop Loss y Take Profit de una posición abierta
TRADE_ACTION_MODIFY	Cambiar los parámetros de una orden comercial anteriormente colocada
TRADE_ACTION_REMOVE	Eliminar una orden comercial pendiente anteriormente colocada
TRADE_ACTION_CLOSE_BY	Cerrar una posición con otra opuesta

ORDER_TYPE_FILLING

Identificador	Descripción
ORDER_FILLING_FOK	Esta política de ejecución establece que la orden puede ser ejecutada exclusivamente en el volumen indicado. Si en el mercado en estos momentos no existe volumen suficiente

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
	de un instrumento financiero, la orden no será ejecutada. El volumen necesario puede componerse de varias ofertas disponibeles en el mercado en el momento actual.
ORDER_FILLING_IOC	Indica que el usuario acepta ejecutar una transacción al mayor volumen alcanzado en el mercado dentro de los límites establecidos en la orden. En caso de que no sea posible la ejecución completa, la orden se ejecutará con el volumen disponible, mientras que el volumen de la orden no ejecutado será cancelado.
ORDER_FILLING_RETURN	Este modo se usa para las órdenes de mercado (ORDER_TYPE_BUY y ORDER_TYPE_SELL), las órdenes límite y las órdenes stop limit (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT y ORDER_TYPE_SELL_STOP_LIMIT) y solo en los modos "Ejecución según mercado" y "Ejecución bursátil". Si una orden de mercado o límite se ejecuta parcialmente con el volumen sobrante, esta no se quitará, sino que seguirá existiendo. Al darse la activación para las órdenes ORDER_TYPE_BUY_STOP_LIMIT y ORDER_TYPE_SELL_STOP_LIMIT, se creará la orden límite correspondiente ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT con el tipo de ejecución ORDER_FILLING_RETURN.

ORDER_TYPE_TIME

Identificador	Descripción
ORDER_TIME_GTC	La orden se encontrará en la cola hasta que sea quitada
ORDER_TIME_DAY	La orden estará activa solo durante el día comercial actual
ORDER_TIME_SPECIFIED	La orden estará activa hasta la fecha de expiración
ORDER_TIME_SPECIFIED_DAY	La orden estará activa hasta las 23:59:59 del día indicado. Si la hora no se encuentra en la sesión comercial, la expiración tendrá lugar en la hora comercial más próxima.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2025, MetaQuotes Ltd.

```
    quit()

# obtenemos la divisa de la cuenta
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# preparamos la estructura de la solicitud
symbol="USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "not found, can not call order_check()")
    mt5.shutdown()
    quit()

# si el símbolo no está disponible en MarketWatch, lo añadimos
if not symbol_info.visible:
    print(symbol, "is not visible, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}}) failed, exit",symbol)
        mt5.shutdown()
        quit()

# preparamos la solicitud
point=mt5.symbol_info(symbol).point
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": 1.0,
    "type": mt5.ORDER_TYPE_BUY,
    "price": mt5.symbol_info_tick(symbol).ask,
    "sl": mt5.symbol_info_tick(symbol).ask-100*point,
    "tp": mt5.symbol_info_tick(symbol).ask+100*point,
    "deviation": 10,
    "magic": 234000,
    "comment": "python script",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# realizamos la comprobación y mostramos el resultado como es
result = mt5.order_check(request)
print(result);
# solicitamos el resultado en forma de diccionario y lo mostramos elemento por element
result_dict=result._asdict()
for field in result_dict.keys():
    print("   {}={}".format(field,result_dict[field]))
    # si se trata de la estructura de una solicitud comercial, también la mostramos el
    if field=="request":
        traderequest_dict=result_dict[field]._asdict()
        for tradereq_filed in traderequest_dict:
            print("       traderequest: {}={}".format(tradereq_filed,traderequest_dict
```

© 2000-2025, MetaQuotes Ltd.

```
# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD
   retcode=0
   balance=101300.53
   equity=68319.53
   profit=-32981.0
   margin=51193.67
   margin_free=17125.86
   margin_level=133.45308121101692
   comment=Done
   request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY', volume=1.0,
       traderequest: action=1
       traderequest: magic=234000
       traderequest: order=0
       traderequest: symbol=USDJPY
       traderequest: volume=1.0
       traderequest: price=108.081
       traderequest: stoplimit=0.0
       traderequest: sl=107.98100000000001
       traderequest: tp=108.181
       traderequest: deviation=10
       traderequest: type=0
       traderequest: type_filling=2
       traderequest: type_time=0
       traderequest: expiration=0
       traderequest: comment=python script
       traderequest: position=0
       traderequest: position_by=0
```

Ver también

order_send, OrderCheck, Tipos de operaciones comerciales, Estructura de la solicitud comercial, Estructura de los resultados de la comprobación de la solicitud comercial, Estructura del resultado de la solicitud comercial

© 2000-2025, MetaQuotes Ltd.

order_send

Envía desde el terminal al servidor comercial una solicitud de ejecución de una operación comercial. La función es un análogo de OrderSend.

```
order_send(
   request      // estructura de la solicitud
   );
```

Parámetros

request

[in] Estructura del tipo MqlTradeRequest que describe la acción comercial necesaria. Parámetro no nombrado obligatorio. Más abajo, podrá ver un ejemplo con el rellenado de una solicitud y la composición de una enumeración.

Valor retornado

El resultado de la ejecución se da en forma de estructura MqlTradeResult. El campo *request* en la respuesta contiene la estructura de la solicitud comercial transmitida a order_send(). En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Estructura de la solicitud comercial MqlTradeRequest

Campo	Descripción
action	Tipo de operación comercial. El valor puede ser uno de los valores de la enumeración TRADE_REQUEST_ACTIONS
magic	Identificador del experto. Permite organizar el procesamiento analítico de órdenes comerciales. Cada experto puede colocar su propio identificador único al enviar una solicitud comercial
order	Ticket de la orden. Es necesario para modificar las órdenes pendientes.
symbol	Nombre del instrumento comercial del que se coloca la orden. No es necesario en las operaciones de modificación de órdenes y el cierre de posiciones
volume	Volumen de la transacción solicitado en lotes. El valor real del volumen al darse la apertura dependerá del tipo de orden según la ejecución.
price	Precio al alcanzar el cual la orden debe ser ejecutada. Para las órdenes de mercado de instrumentos con el tipo de ejecución "Market Execution" (SYMBOL_TRADE_EXECUTION_MARKET) que tengan el tipo TRADE_ACTION_DEAL no se requerirá indicar el precio
stoplimit	Precio al que se colocará una orden límite pendiente cuando el precio alcance el valor price (esta condición es obligatoria). Hasta ese momento, la orden pendiente no se mostrará en el sistema comercial
sl	Precio al que se activa una orden Stop Loss cuando el precio se mueve en una dirección no rentable
tp	Precio al que se activa una orden Take Profit cuando el precio se mueve en una dirección rentable

© 2000-2025, MetaQuotes Ltd.

Campo	Descripción
deviation	Desviación máxima permitida respecto al precio solicitado, establecida en puntos
type	Tipo de orden. El valor puede ser uno de los valores de la enumeración ORDER_TYPE
type_filling	Tipo de orden según su ejecución. El valor puede ser uno de los valores ORDER_TYPE_FILLING
type_time	Tipo de orden según su expiración. El valor puede ser uno de los valores ORDER_TYPE_TIME
expiration	Plazo de expiración de la orden pendiente (para las órdenes del tipo TIME_SPECIFIED)
comment	Comentarios a la orden
position	Ticket de la posición. Se debe rellenar al cambiar y cerrar una posición, para posibilitar su identificación unívoca. Normalmente, se corresponde con el ticket de la orden que ha dado lugar a la apertura de la posición.
position_by	Ticket de la posición opuesta. Se usa al cerrar una posición mediante otra opuesta, abierta para el mismo instrumento, pero en dirección contraria.

Observación

La solicitud comercial pasar por varios estadios de comprobación en el servidor comercial. En primer lugar, se comprueba que los campos de la solicitud *request* se hayan rellenado correctamente, y si no hay errores, el servidor aceptará la orden para su posterior procesamiento. Podrá ver información detallada sobre la ejecución de operaciones comerciales en la descripción de la función OrderSend.

Ejemplo:

```
import time
import MetaTrader5 as mt5

# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ", mt5.__author__)
print("MetaTrader5 package version: ", mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# preparamos la estructura de la solicitud de compra
symbol = "USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "not found, can not call order_check()")
    mt5.shutdown()
    quit()
```

© 2000-2025, MetaQuotes Ltd.

```
# si el símbolo no está disponible en MarketWatch, lo añadimos
if not symbol_info.visible:
    print(symbol, "is not visible, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}}) failed, exit",symbol)
        mt5.shutdown()
        quit()

lot = 0.1
point = mt5.symbol_info(symbol).point
price = mt5.symbol_info_tick(symbol).ask
deviation = 20
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_BUY,
    "price": price,
    "sl": price - 100 * point,
    "tp": price + 100 * point,
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script open",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# enviamos la solicitud comercial
result = mt5.order_send(request)
# comprobamos el resultado de la ejecución
print("1. order_send(): by {} {} lots at {} with deviation={} points".format(symbol,lo
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("2. order_send failed, retcode={}".format(result.retcode))
   # solicitamos el resultado en forma de diccionario y lo mostramos elemento por eler
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # si se trata de la estructura de una solicitud comercial, también la mostramo
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_
    print("shutdown() and quit")
    mt5.shutdown()
    quit()

print("2. order_send done, ", result)
print("   opened position with POSITION_TICKET={}".format(result.order))
```

© 2000-2025, MetaQuotes Ltd.

```
print("   sleep 2 seconds before closing position #{}".format(result.order))
time.sleep(2)
# creamos una solicitud de cierre
position_id=result.order
price=mt5.symbol_info_tick(symbol).bid
deviation=20
request={
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_SELL,
    "position": position_id,
    "price": price,
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script close",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}
# enviamos la solicitud comercial
result=mt5.order_send(request)
# comprobamos el resultado de la ejecución
print("3. close position #{}: sell {} {} lots at {} with deviation={} points".format(p
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("4. order_send failed, retcode={}".format(result.retcode))
    print("   result",result)
else:
    print("4. position #{} closed, {}".format(position_id,result))
   # solicitamos el resultado en forma de diccionario y lo mostramos elemento por eler
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # si se trata de la estructura de una solicitud comercial, también la mostramo
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
```

Resultado

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
1. order_send(): by USDJPY 0.1 lots at 108.023 with deviation=20 points
2. order_send done,  OrderSendResult(retcode=10009, deal=535084512, order=557416535, v
   opened position with POSITION_TICKET=557416535
   sleep 2 seconds before closing position #557416535
3. close position #557416535: sell USDJPY 0.1 lots at 108.018 with deviation=20 points
```

© 2000-2025, MetaQuotes Ltd.

```
4. position #557416535 closed, OrderSendResult(retcode=10009, deal=535084631, order=55
   retcode=10009
   deal=535084631
   order=557416654
   volume=0.1
   price=108.015
   bid=108.015
   ask=108.02
   comment=Request executed
   request_id=55
   retcode_external=0
   request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY', volume=0.1,
       traderequest: action=1
       traderequest: magic=234000
       traderequest: order=0
       traderequest: symbol=USDJPY
       traderequest: volume=0.1
       traderequest: price=108.018
       traderequest: stoplimit=0.0
       traderequest: sl=0.0
       traderequest: tp=0.0
       traderequest: deviation=20
       traderequest: type=1
       traderequest: type_filling=2
       traderequest: type_time=0
       traderequest: expiration=0
       traderequest: comment=python script close
       traderequest: position=557416535
       traderequest: position_by=0
```

Ver también

order_check, OrderSend,Tipos de operaciones comerciales, Estructura de la solicitud comercial, Estructura de los resultados de la comprobación de la solicitud comercial, Estructura del resultado de la solicitud comercial

© 2000-2025, MetaQuotes Ltd.

positions_total

Obtiene el número de posiciones abiertas.

```
positions_total()
```

Valor retornado

Valor de tipo entero.

Observación

La función es un análogo de PositionsTotal.

Ejemplo:

```
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# comprobamos la presencia de posiciones abiertas
positions_total=mt5.positions_total()
if positions_total>0:
    print("Total positions=",positions_total)
else:
    print("Positions not found")

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
```

Ver también

positions_get, orders_total

© 2000-2025, MetaQuotes Ltd.

positions_get

Obtiene las posiciones abiertas con posibilidad de filtrado según un símbolo o ticket. Existen 3 variantes de llamada.

Llamada sin parámetros. Retorna las posiciones abiertas de todos los símbolos.

```
positions_get()
```

Llamada con indicación del símbolo del que debemos obtener las posiciones abiertas.

```
positions_get(
   symbol="SYMBOL"       // nombre del símbolo
)
```

Llamada con indicación del grupo de símbolos de los que debemos obtener las posiciones abiertas.

```
positions_get(
   group="GROUP"         // filtro de selección de posiciones según los símbolos
)
```

Llamada con indicación del ticket de la posición.

```
positions_get(
   ticket=TICKET         // ticket
)
```

Parámetros

symbol="SYMBOL"

[in] Nombre del símbolo. Parámetro nombrado no obligatorio. Si se ha indicado el símbolo, el parámetro *ticket* será ignorado.

group="GROUP"

[in] Filtro para seleccionar un grupo solo con los símbolos necesarios. Parámetro nombrado no obligatorio. Si el grupo ha sido establecido, la función retornará solo las posiciones que cumplan con el criterio establecido para el nombre del símbolo.

ticket=TICKET

[in] Ticket de la posición (POSITION_TICKET). Parámetro nombrado no obligatorio.

Valor retornado

Retorna la información como estructura de tuplas nombradas (namedtuple). En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función permite obtener en una sola llamada todas las posiciones abiertas; sería un análogo de la unión de PositionsTotal y PositionSelect.

El parámetro *group* puede contener varias condiciones separadas por comas. Las condiciones se pueden establecer como máscara con el uso de '*'. Para realizar exclusiones, se puede usar el símbolo de negación lógica '!'. En este caso, además, todas las condiciones se aplican de forma secuencial, es decir, primero se deben indicar las inclusiones en el grupo, y después las condiciones de exclusión. Por ejemplo, *group*="*, !EUR" significa que primero debemos seleccionar las posiciones

© 2000-2025, MetaQuotes Ltd.

según todos los símbolos, y después excluir aquellos de ellos que contengan en el nombre del símbolo "EUR".

Ejemplo:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # cuántas columnas mostramos
pd.set_option('display.width', 1500)      # máx. anchura del recuadro para la muestra
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtenemos las posiciones abiertas de USDCHF
positions=mt5.positions_get(symbol="USDCHF")
if positions==None:
    print("No positions on USDCHF, error code={}".format(mt5.last_error()))
elif len(positions)>0:
    print("Total positions on USDCHF =",len(positions))
    # mostramos todas las posiciones abiertas
    for position in positions:
        print(position)

# obtenemos la lista de posiciones en los símbolos cuyos nombres contienen "*USD*"
usd_positions=mt5.positions_get(group="*USD*")
if usd_positions==None:
    print("No positions with group=\"*USD*\", error code={}".format(mt5.last_error()))
elif len(usd_positions)>0:
    print("positions_get(group=\"*USD*\")={}".format(len(usd_positions)))
    # mostramos estas posiciones en forma de recuadro con la ayuda de pandas.DataFrame
    df=pd.DataFrame(list(usd_positions),columns=usd_positions[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    df.drop(['time_update', 'time_msc', 'time_update_msc', 'external_id'], axis=1, inp
    print(df)

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

positions_get(group="*USD*")=5
```

© 2000-2025, MetaQuotes Ltd.

	ticket	time	type	magic	identifier	reason	volume	price_open
0	548297723	2020-03-18 15:00:55	1	0	548297723	3	0.01	1.09301
1	548655158	2020-03-18 20:31:26	0	0	548655158	3	0.01	1.08676
2	548663803	2020-03-18 20:40:04	0	0	548663803	3	0.01	1.08640
3	548847168	2020-03-19 01:10:05	0	0	548847168	3	0.01	1.09545
4	548847194	2020-03-19 01:10:07	0	0	548847194	3	0.02	1.09536

Ver también

positions_total, orders_get

© 2000-2025, MetaQuotes Ltd.

history_orders_total

Obtiene el número de órdenes en la historia comercial en el intervalo indicado.

```
history_orders_total(
   date_from,     // fecha a partir de la cual se solicitan las órdenes
   date_to        // fecha hasta la cual se solicitan las órdenes
   )
```

Parámetros

date_from

[in] Fecha a partir de la cual se solicitan las órdenes. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio.

date_to

[in] Fecha hasta la cual se solicitan las órdenes. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio.

Valor retornado

Valor de tipo entero.

Observación

La función es un análogo de HistoryOrdersTotal.

Ejemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtenemos el número de órdenes en la historia
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_total(from_date, datetime.now())
if history_orders>0:
    print("Total history orders=",history_orders)
else:
    print("Orders not found in history")

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

Ver también

history_orders_get, history_deals_total

© 2000-2025, MetaQuotes Ltd.

history_orders_get

Obtiene las órdenes de la historia comercial con posibilidad de filtrado según un ticket o posición. Existen 3 variantes de llamada.

Llamada con indicación del intervalo temporal. Retorna todas las órdenes que entran en el intervalo establecido.

```
history_orders_get(
   date_from,           // fecha a partir de la cual se solicitan las órdenes
   date_to,             // fecha hasta la cual se solicitan las órdenes
   group="GROUP"        // filtro de selección de órdenes según los símbolos
   )
```

Llamada con indicación del ticket de la orden. Retorna la orden con el ticket indicado.

```
history_orders_get(
   ticket=TICKET        // ticket de la orden
)
```

Llamada con indicación del ticket de la posición. Retorna todas las órdenes que tienen en la propiedad ORDER_POSITION_ID el ticket de la posición indicado.

```
history_orders_get(
   position=POSITION    // ticket de la posición
)
```

Parámetros

date_from

[in] Fecha a partir de la cual se solicitan las órdenes. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio, se indica en primer lugar.

date_to

[in] Fecha hasta la cual se solicitan las órdenes. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio, se indica en segundo lugar.

group="GROUP"

[in] Filtro para seleccionar un grupo solo con los símbolos necesarios. Parámetro nombrado no obligatorio. Si el grupo ha sido establecido, la función retornará las órdenes que cumplan con el criterio establecido para el nombre del símbolo.

ticket=TICKET

[in] Ticket de la orden que se debe obtener. Parámetro no obligatorio. Si el ticket de la orden no ha sido indicado, el filtro no se usará.

position=POSITION

[in] Ticket de la posición (se guarda en ORDER_POSITION_ID) para la que debemos obtener todas las órdenes. Parámetro no obligatorio. Si el ticket de la posición no ha sido indicado, el filtro no se usará.

Valor retornado

© 2000-2025, MetaQuotes Ltd.

Retorna la información como estructura de tuplas nombradas (namedtuple). En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función permite obtener en una sola llamada todas las órdenes históricas en el periodo indicado; sería un análogo de la unión de HistoryOrdersTotal y HistoryOrderSelect.

El parámetro *group* puede contener varias condiciones separadas por comas. Las condiciones se pueden establecer como máscara con el uso de '*'. Para realizar exclusiones, se puede usar el símbolo de negación lógica '!'. En este caso, además, todas las condiciones se aplican de forma secuencial, es decir, primero se deben indicar las inclusiones en el grupo, y después las condiciones de exclusión. Por ejemplo, *group*="*, !EUR" significa que primero debemos seleccionar las transacciones según todos los símbolos, y después excluir aquellos de ellos que contengan en el nombre del símbolo "EUR".

Ejemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # cuántas columnas mostramos
pd.set_option('display.width', 1500)      # máx. anchura del recuadro para la muestra
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtenemos el número de órdenes en la historia
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_get(from_date, to_date, group="*GBP*")
if history_orders==None:
    print("No history orders with group=\"*GBP*\", error code={}".format(mt5.last_erro
elif len(history_orders)>0:
    print("history_orders_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,le
print()

# mostramos todas las órdenes históricas según el ticket de una posición
position_id=530218319
position_history_orders=mt5.history_orders_get(position=position_id)
if position_history_orders==None:
    print("No orders with position #{}".format(position_id))
    print("error code =",mt5.last_error())
elif len(position_history_orders)>0:
    print("Total history orders on position #{}: {}".format(position_id,len(position_h
   # mostramos todas las órdenes históricas que tienen el ticket de posición indicado
```

© 2000-2025, MetaQuotes Ltd.

```
    for position_order in position_history_orders:
        print(position_order)
    print()
    # mostramos estas órdenes en forma de recuadro con la ayuda de pandas.DataFrame
    df=pd.DataFrame(list(position_history_orders),columns=position_history_orders[0]._
    df.drop(['time_expiration','type_time','state','position_by_id','reason','volume_
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    df['time_done'] = pd.to_datetime(df['time_done'], unit='s')
    print(df)

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

history_orders_get(2020-01-01 00:00:00, 2020-03-25 17:17:32.058795, group="*GBP*")=14

Total history orders on position #530218319: 2
TradeOrder(ticket=530218319, time_setup=1582282114, time_setup_msc=1582282114681, time
TradeOrder(ticket=535548147, time_setup=1583176242, time_setup_msc=1583176242265, time

      ticket           time_setup  time_setup_msc           time_done  time_done_msc  t
0  530218319 2020-02-21 10:48:34   1582282114681 2020-02-21 16:49:37  1582303777582
1  535548147 2020-03-02 19:10:42   1583176242265 2020-03-02 19:10:42  1583176242265
```

Ver también

history_deals_total, history_deals_get

© 2000-2025, MetaQuotes Ltd.

history_deals_total

Obtiene el número de transacciones en la historia comercial en el intervalo indicado.

```
history_deals_total(
   date_from,     // fecha a partir de la cual se solicitan las transacciones
   date_to        // fecha hasta la cual se solicitan las transacciones
   )
```

Parámetros

date_from

[in] Fecha a partir de la cual se solicitan las transacciones. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio.

date_to

[in] Fecha hasta la cual se solicitan las transacciones. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio.

Valor retornado

Valor de tipo entero.

Observación

La función es un análogo de HistoryDealsTotal.

Ejemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtenemos el número de transacciones en la historia
from_date=datetime(2020,1,1)
to_date=datetime.now()
deals=mt5.history_deals_total(from_date, to_date)
if deals>0:
    print("Total deals=",deals)
else:
    print("Deals not found in history")

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

Ver también

history_deals_get, history_orders_total

© 2000-2025, MetaQuotes Ltd.

history_deals_get

Obtiene las transacciones de la historia comercial en el intervalo indicado con posibilidad de filtrado según un ticket o posición.

Llamada con indicación del intervalo temporal. Retorna todas las transacciones que entran en el intervalo establecido.

```
history_deals_get(
   date_from,           // fecha a partir de la cual se solicitan las transacciones
   date_to,             // fecha hasta la cual se solicitan los ticks
   group="GROUP"        // filtro de selección de transacciones según los símbolos
   )
```

Llamada con indicación del ticket de la orden. Retorna todas las transacciones que tienen el ticket de orden indicado en la propiedad DEAL_ORDER.

```
history_deals_get(
   ticket=TICKET        // ticket de la orden
)
```

Llamada con indicación del ticket de la posición. Retorna todas las transacciones que tienen el ticket de posición indicado en la propiedad DEAL_POSITION_ID.

```
history_deals_get(
   position=POSITION    // ticket de la posición
)
```

Parámetros

date_from

[in] Fecha a partir de la cual se solicitan las órdenes. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio, se indica en primer lugar.

date_to

[in] Fecha hasta la cual se solicitan las órdenes. Se establece con el objeto datetime o como número de segundos transcurridos desde el 1970.01.01. Parámetro no nombrado obligatorio, se indica en segundo lugar.

group="GROUP"

[in] Filtro para seleccionar un grupo solo con los símbolos necesarios. Parámetro nombrado no obligatorio. Si el grupo ha sido establecido, la función retornará las transacciones que cumplan con el criterio establecido para el nombre del símbolo.

ticket=TICKET

[in] Ticket de la orden (se guarda en DEAL_ORDER) para la que debemos obtener todas las transacciones. Parámetro no obligatorio. Si el ticket de la orden no ha sido indicado, el filtro no se usará.

position=POSITION

[in] Ticket de la posición (se guarda en DEAL_POSITION_ID) para la que debemos obtener todas las transacciones. Parámetro no obligatorio. Si el ticket de la posición no ha sido indicado, el filtro no se usará.

© 2000-2025, MetaQuotes Ltd.

Valor retornado

Retorna la información como estructura de tuplas nombradas (namedtuple). En caso de error, retorna None; la información sobre el error se puede obtener con la ayuda de last_error().

Observación

La función permite obtener en una sola llamada todas las transacciones de la historia en el periodo indicado; sería un análogo de la unión de HistoryDealsTotal y HistoryDealSelect.

El parámetro *group* permite filtrar las transacciones por símbolos. Está permitido usar '*' al inicio y el final de la línea.

El parámetro *group* puede contener varias condiciones separadas por comas. Las condiciones se pueden establecer como máscara con el uso de '*'. Para realizar exclusiones, se puede usar el símbolo de negación lógica '!'. En este caso, además, todas las condiciones se aplican de forma secuencial, es decir, primero se deben indicar las inclusiones en el grupo, y después las condiciones de exclusión. Por ejemplo, *group*="*, !EUR" significa que primero debemos seleccionar las transacciones según todos los símbolos, y después excluir aquellos de ellos que contengan en el nombre del símbolo "EUR".

Ejemplo:

```
import MetaTrader5 as mt5
from datetime import datetime
import pandas as pd
pd.set_option('display.max_columns', 500) # cuántas columnas mostramos
pd.set_option('display.width', 1500)      # máx. anchura del recuadro para la muestra
# mostramos los datos sobre el paquete MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# establecemos la conexión con el terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtenemos el número de transacciones en la historia
from_date=datetime(2020,1,1)
to_date=datetime.now()
# obtenemos en el intervalo indicado las transacciones de los símbolos cuyos nombres c
deals=mt5.history_deals_get(from_date, to_date, group="*GBP*")
if deals==None:
    print("No deals with group=\"*USD*\", error code={}".format(mt5.last_error()))
elif len(deals)> 0:
    print("history_deals_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,len

# obtenemos las transacciones de los símbolos cuyos nombres no contienen ni "EUR" ni '
deals = mt5.history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*")
if deals == None:
    print("No deals, error code={}".format(mt5.last_error()))
elif len(deals) > 0:
```

© 2000-2025, MetaQuotes Ltd.

```
    print("history_deals_get(from_date, to_date, group=\"*,!*EUR*,!*GBP*\") =", len(de
    # mostramos todos los datos obtenidos como son
    for deal in deals:
        print("  ",deal)
    print()
    # mostramos todas las transacciones en forma de recuadro con la ayuda de pandas.Da
    df=pd.DataFrame(list(deals),columns=deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)
print("")

# obtenemos todas las transacciones relacionadas con la posición #530218319
position_id=530218319
position_deals = mt5.history_deals_get(position=position_id)
if position_deals == None:
    print("No deals with position #{}".format(position_id))
    print("error code =", mt5.last_error())
elif len(position_deals) > 0:
    print("Deals with position id #{}: {}".format(position_id, len(position_deals)))
    # mostramos todas las transacciones en forma de recuadro con la ayuda de pandas.Da
    df=pd.DataFrame(list(position_deals),columns=position_deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)

# finalizamos la conexión con el terminal MetaTrader 5
mt5.shutdown()

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

history_deals_get(from_date, to_date, group="*GBP*") = 14

history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*") = 7
   TradeDeal(ticket=506966741, order=0, time=1582202125, time_msc=1582202125419, type=
   TradeDeal(ticket=507962919, order=530218319, time=1582303777, time_msc=158230377758
   TradeDeal(ticket=513149059, order=535548147, time=1583176242, time_msc=158317624226
   TradeDeal(ticket=516943494, order=539349382, time=1583510003, time_msc=158351000389
   TradeDeal(ticket=516943915, order=539349802, time=1583510025, time_msc=158351002505
   TradeDeal(ticket=517139682, order=539557870, time=1583520201, time_msc=158352020122
   TradeDeal(ticket=517139716, order=539557909, time=1583520202, time_msc=158352020297

      ticket      order                time       time_msc  type  entry  magic  positi
0  506966741          0 2020-02-20 12:35:25  1582202125419     2      0      0
1  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0    5302
2  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    5302
3  516943494  539349382 2020-03-06 15:53:23  1583510003895     1      0      0    5393
4  516943915  539349802 2020-03-06 15:53:45  1583510025054     0      0      0    5393
5  517139682  539557870 2020-03-06 18:43:21  1583520201227     0      1      0    5393
```

© 2000-2025, MetaQuotes Ltd.

```
6  517139716  539557909 2020-03-06 18:43:22  1583520202971     1      1      0    5393

Deals with position id #530218319: 2
      ticket      order                time       time_msc  type  entry  magic  positi
0  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0    5302
1  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    5302
```

Ver también

history_deals_total, history_orders_get

© 2000-2025, MetaQuotes Ltd.

Modelos ONNX en el aprendizaje automático

ONNX (Open Neural Network Exchange) es una biblioteca de código abierto para construir redes neuronales de aprendizaje profundo. Este proyecto tiene algunas ventajas importantes:.

- El soporte para ONNX se realiza a través de grandes compañías como Microsoft, Facebook, Amazon y otros socios..
- El formato abierto permite convertir formatos entre diferentes herramientas de aprendizaje automático, ONNXMLTools de Microsoft permite convertir modelos al formato ONNX..
- MQL5 posibilta la conversión automática de los datos de entrada y salida del modelo si el tipo de parámetro transmitido no coincide con el modelo.
- Es posible crear modelos ONNX con una gran variedad de herramientas de aprendizaje automático, actualmente tienen soporte en Caffe2, Microsoft Cognitive Toolkit, MXNet, PyTorch y OpenCV, y también hay interfaces para muchos otros frameworks y bibliotecas populares.
- El lenguaje MQL5 permite implementar el modelo ONNX en una estrategia comercial y usar todas las ventajas de la plataforma MetaTrader 5 para trabajar de forma eficiente en los mercados financieros.
- Antes de iniciar el modelo en operaciones reales, puede probar el modelo con datos históricos en el simulador de estrategias sin utilizar herramientas de terceros.

Para trabajar con ONNX, MQL5 ofrece las siguientes funciones:

Función	Acción
OnnxCreate	Crea una sesión de ONNX cargando el modelo desde un archivo *.onnx
OnnxCreateFromBuffer	Crea una sesión de ONNX cargando el modelo desde un array de datos
OnnxRelease	Eliminar sesión ONNX
OnnxRun	Iniciar un modelo ONNX para su ejecución
OnnxGetInputCount	Obtener el número de parámetros de entrada del modelo ONNX
OnnxGetOutputCount	Obtener el número de parámetros de salida del modelo ONNX
OnnxGetInputName	Obtener el nombre del parámetro de entrada del modelo según el índice
OnnxGetOutputName	Obtener el nombre de un parámetro de salida del modelo según el índice
OnnxGetInputTypeInfo	Obtener la descripción del tipo de parámetro de entrada del modelo
OnnxGetOutputTypeInfo	Obtener la descripción del tipo de parámetro de salida del modelo
OnnxSetInputShape	Establece la dimensionalidad de los datos de entrada del modelo según el índice
OnnxSetOutputShape	Establece la dimensionalidad de los datos de salida del modelo según el índice

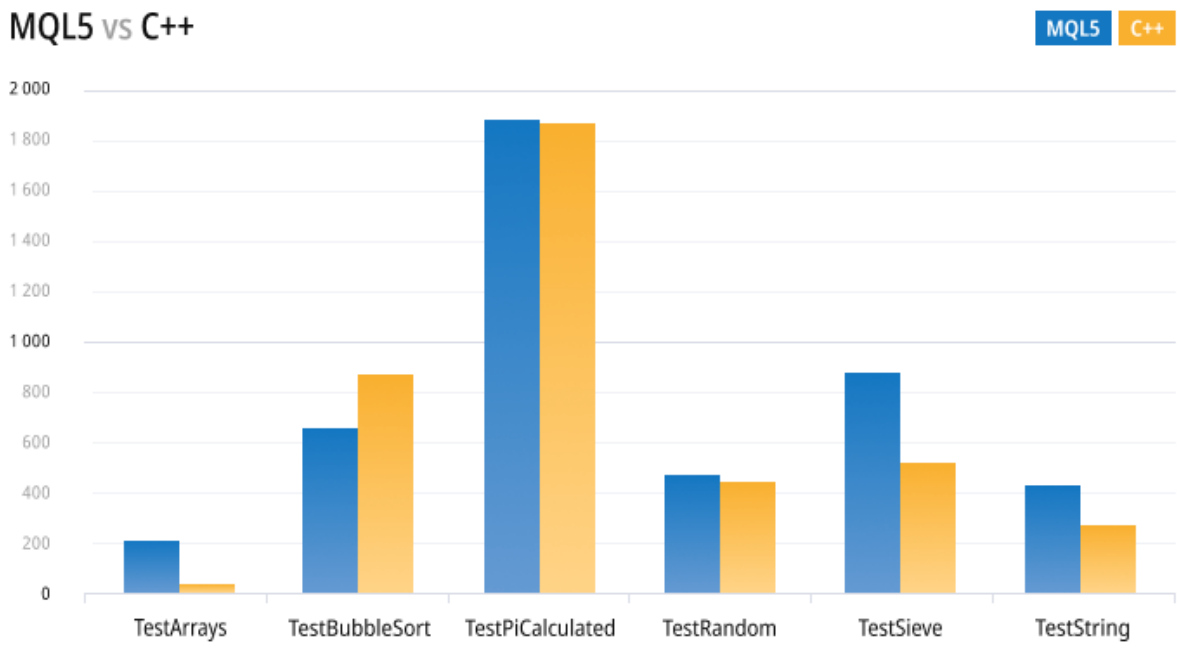
© 2000-2025, MetaQuotes Ltd.

Soporte de ONNX en MQL5

ONNX es un formato abierto creado para representar modelos de aprendizaje automático. Este estándar define un conjunto común de declaraciones y un formato de archivos que permite a los desarrolladores usar modelos en diferentes plataformas y entornos de ejecución, utilizando diferentes herramientas y compiladores.

Por lo tanto, el formato abierto de ONNX permite obtener y desplazar modelos de aprendizaje automático entre diferentes plataformas y herramientas de aprendizaje automático. Por eso se ha añadido el soporte de ONNX al lenguaje MQL5: los desarrolladores de IA podrá así ejecutar los modelos creados en un entorno de ejecución de alto rendimiento en la plataforma MetaTrader 5.

El lenguaje MQL5 no resulta inferior en velocidad de ejecución a las aplicaciones C++. Los resultados de la ejecución de las pruebas estándar en MQL5 y C++ lo confirman. Cuanto más baja sea la columna, menos tiempo se dedicará a la ejecución, y mejor será el resultado (tiempo en milisegundos). Las pruebas se han realizado en Windows 10 (build 17763) x64, Xeon E5-2630 v4 @ 2.20GHz, Memory: 65457 Mb.



Con las nuevas operaciones comerciales asincrónicas y el soporte nativo de ONNX, se abren nuevas oportunidades que antes solo estaban disponibles para profesionales selectos y tráders institucionales. El soporte de ONNX en MQL5 permite a los tráders entrenar modelos para mercados financieros en su entorno de desarrollo preferido y luego operar con bajos costes de red, alta velocidad de actualización de la profundidad de mercado y uso de envío asíncrono de órdenes.

Actualmente, ONNX está siendo desarrollado y mantenido de forma conjunta por empresas como Microsoft, Facebook, Amazon y otros socios, lo cual garantiza el mayor desarrollo de este proyecto de código abierto.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Conversión de formatos

El formato ONNX está abierto, lo cual permite guardar en él modelos de varias herramientas de aprendizaje automático. Este formato es compatible con muchas plataformas, incluidas Chainer, Caffee2 y PyTorch.

Una de las herramientas más populares para convertir modelos al formato ONNX es ONNXMLTools de Microsoft.

Las instrucciones para instalar y usar ONNXMLTools están disponibles en el repositorio en GitHub. Actualmente se admiten los siguientes conjuntos de herramientas:

- Keras (envoltorio del convertidor keras2onnx)
- Tensorflow (envoltorio del convertidor tf2onnx)
- scikit-learn (envoltorio del convertidor skl2onnx)
- Apple Core ML
- Spark ML (modo experimental)
- LightGBM
- libscm;
- XGBoost;
- H2O
- CatBoost

La instalación de ONNXMLTools es simple y se describe en la página del proyecto https://github.com/onnx/onnxmltools#install, ahí también se ofrecen algunos ejemplos de conversión de modelos.

© 2000-2025, MetaQuotes Ltd.

Autoconversión de los valores input y output al iniciar los modelos ONNX

En la actual versión de ONNX en el lenguaje MQL5, como valores input/output se soportan solo tensores, los arrays de datos con uno de los siguientes tipos para los elementos:

Tipo ONNX	Se corresponde con el tipo MQL5
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_UINT8	uchar
ONNX_DATA_TYPE_INT8	char
ONNX_DATA_TYPE_UINT16	ushort
ONNX_DATA_TYPE_INT16	short
ONNX_DATA_TYPE_INT32	int
ONNX_DATA_TYPE_INT64	long
ONNX_DATA_TYPE_FLOAT16	—
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	uint
ONNX_DATA_TYPE_UINT64	ulong
ONNX_DATA_TYPE_COMPLEX64	—
ONNX_DATA_TYPE_COMPLEX128	complex
ONNX_DATA_TYPE_BFLOAT16	—
ONNX_DATA_TYPE_STRING	—

Como valores input/output en los modelos ONNX se pueden transmitir solo arrays, vectores o matrices (en lo sucesivo **Datos**).

Si el tipo del parámetro transmitido no coincide con el tipo del parámetro del modelo ONNX y al llamar a OnnxRun no se especifica la bandera ONNX_NO_CONVERSION, se utilizará la autoconversión. La autoconversión implica que antes de ejecutar el modelo ONNX, los **Datos** de usuario se copiarán a tensores ONNX con conversión.

Al ejecutar el modelo ONNX sin autoconversión, el modelo ONNX se calculará con los **Datos** sin ningún copiado adicional.

> **¡IMPORTANTE!** La autoconversión no controla el desbordamiento (truncate), por lo que deberá monitorear cuidadosamente tanto los datos como sus tipos transmitidos al modelo ONNX.

La autoconversión admite los siguientes tipos de ONNX:

© 2000-2025, MetaQuotes Ltd.

- ONNX_DATA_TYPE_BOOL
- ONNX_DATA_TYPE_FLOAT
- ONNX_DATA_TYPE_UINT8
- ONNX_DATA_TYPE_INT8
- ONNX_DATA_TYPE_UINT16
- ONNX_DATA_TYPE_INT16
- ONNX_DATA_TYPE_INT32
- ONNX_DATA_TYPE_INT64
- ONNX_DATA_TYPE_FLOAT16
- ONNX_DATA_TYPE_DOUBLE
- ONNX_DATA_TYPE_UINT32
- ONNX_DATA_TYPE_UINT64
- ONNX_DATA_TYPE_COMPLEX64
- ONNX_DATA_TYPE_COMPLEX128

Datos no soportados:

- ONNX_DATA_TYPE_BFLOAT16
- ONNX_DATA_TYPE_STRING

Reglas de conversión automática por tipos de tensor

Si el tipo MQL5 no está incluido en la lista de tipos soportados por el modelo, la ejecución del modelo ONNX finalizará con el error ERR_ONNX_NOT_SUPPORTED (código de error 5802).

Nota: al realizar la autoconversión, el tipo color se trata como uint, mientras que el tipo datetime se trata como long.

Autoconversión de valores input

Tipo ONNX (tipo de elemento del tensor)	Tipo MQL5 compatible con la autoconversión
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex Al realizar la conversión, los elementos de los **Datos** se comprueban mediante una comparación simple con cero
ONNX_DATA_TYPE_FLOAT16	float, double
ONNX_DATA_TYPE_FLOAT	char, uchar, short, ushort, int, color, uint, datetime, long, ulong, float, double
ONNX_DATA_TYPE_UINT8	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	ver ONNX_DATA_TYPE_FLOAT

© 2000-2025, MetaQuotes Ltd.

Tipo ONNX (tipo de elemento del tensor)	Tipo MQL5 compatible con la autoconversión
ONNX_DATA_TYPE_UINT16	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

Autoconversión de valores output

Tipo ONNX (tipo de elemento del tensor)	Tipo MQL5 compatible con la autoconversión
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex Si el elemento del tensor es cero, para el elemento de los Datos se establecerá el valor 0, de lo contrario, 1
ONNX_DATA_TYPE_FLOAT16	float, double
ONNX_DATA_TYPE_FLOAT	char, uchar, short, ushort, int, color, uint, datetime, long, ulong, float, double
ONNX_DATA_TYPE_UINT8	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT16	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	ver ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

© 2000-2025, MetaQuotes Ltd.

Ver también

Conversión de tipos

© 2000-2025, MetaQuotes Ltd.

Creación de un modelo

Hay varias formas de obtener un modelo terminado en formato ONNX. La biblioteca reconocida es ONNX Model Zoo, que contiene varios modelos ONNX previamente entrenados para diferentes tipos de tareas. La ventaja de esta colección es que el notebook de cada modelo contiene enlaces al conjunto de datos de entrenamiento, además de enlaces al documento científico original que describe la arquitectura del modelo.

La mayoría de plataformas para el aprendizaje automático usa Python, para instalar el entorrno ONNX para Python, use uno de los comandos siguientes:

```
pip install onnxruntime        # CPU build
pip install onnxruntime-gpu    # GPU build
```

Para invocar el tiempo de ejecución de ONNX en Python, use el siguiente comando

```
import onnxruntime
session = onnxruntime.InferenceSession("path to model")
```

Los datos de entrada y salida para usar el modelo deben encontrarse en la documentación del modelo correspondiente. También puede usar recursos de visualización como Netron o WinML Dashboard para ver el modelo. El tiempo de ejecución de ONNX también permite consultar los metadatos del modelo y sus datos de entrada y salida:

```
results = session.run(["output1", "output2"], {
                      "input1": indata1, "input2": indata2})
results = session.run([], {"input1": indata1, "input2": indata2})
```

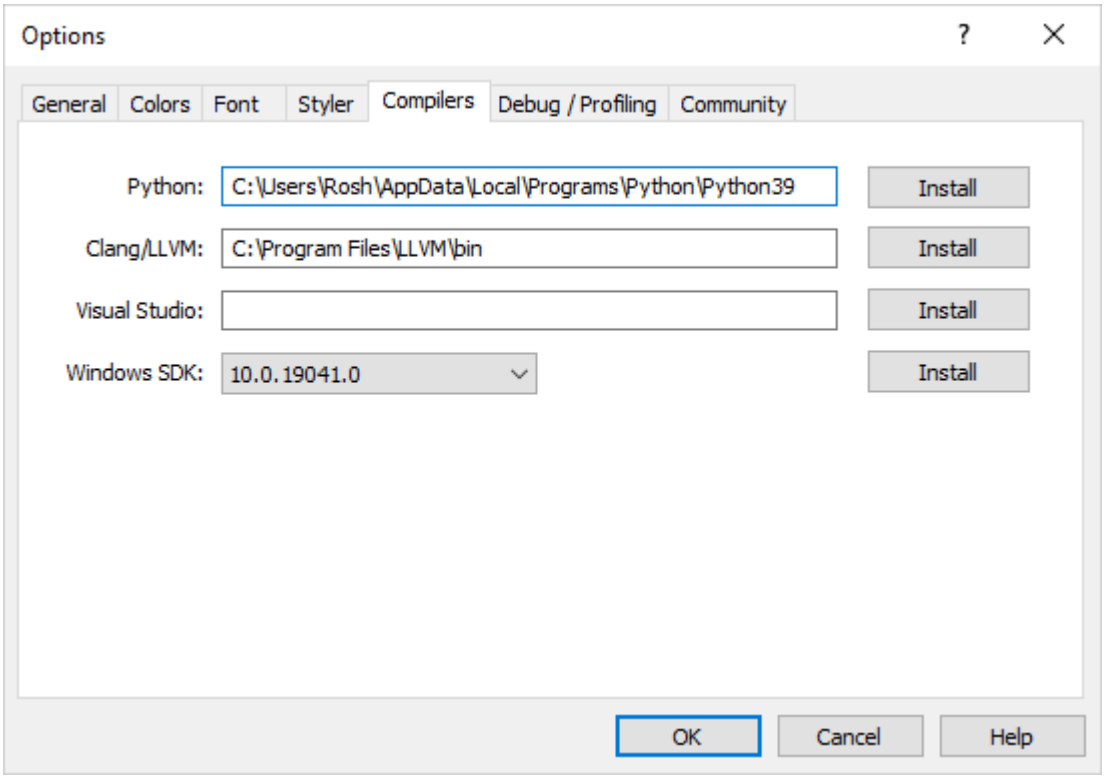
Puede crear un modelo ONNX directamente en el terminal MetaTrader 5 o en el entorno de trabajo del MetaEditor en Python.

Python в MetaTrader 5

La plataforma MetaTrader 5 ofrece soporte al trabajo con scripts de Python listos para usar. Para ello, los desarrolladores del terminal ofrecen y mantienen un módulo MetaTrader5 para Python — https://pypi.org/project/MetaTrader5.

El entorno de desarrollo integrado de MetaEditor permite escribir no solo aplicaciones en MQL5, sino también ejecutar scripts de Python directamente desde el editor. Para hacer esto, debe especificar la ruta al archivo ejecutable en los ajustes del MetaEditor:

© 2000-2025, MetaQuotes Ltd.



Si no tiene Python en su computadora, pulse "Instalar" para descargar el archivo de instalación.

Puede crear un script de Python en el MetaEdtior o cargarlo en la carpeta de datos del terminal y ejecutarlo directamente usando la tecla F7 ("Compilar"). Después de eso, se abrirá el terminal MetaTrader 5 y se iniciará el script en el gráfico actual. Los mensajes de la consola Python (stdout, stderr) se mostrarán en el apartado "Errores".

Esquema de trabajo con modelos en MetaTrader 5

El lenguaje MQL5 le permite ejecutar modelos ONNX directamente en el terminal MetaTrader 5. Esto se hace en 3 pasos:

1. Entrene el modelo en una plataforma de terceros, como Python.
2. Convierta modelo a ONNX
3. Incluya el modelo ONNX en el asesor utilizando las funciones ONNX y ejecútelo en el terminal MetaTrader 5.

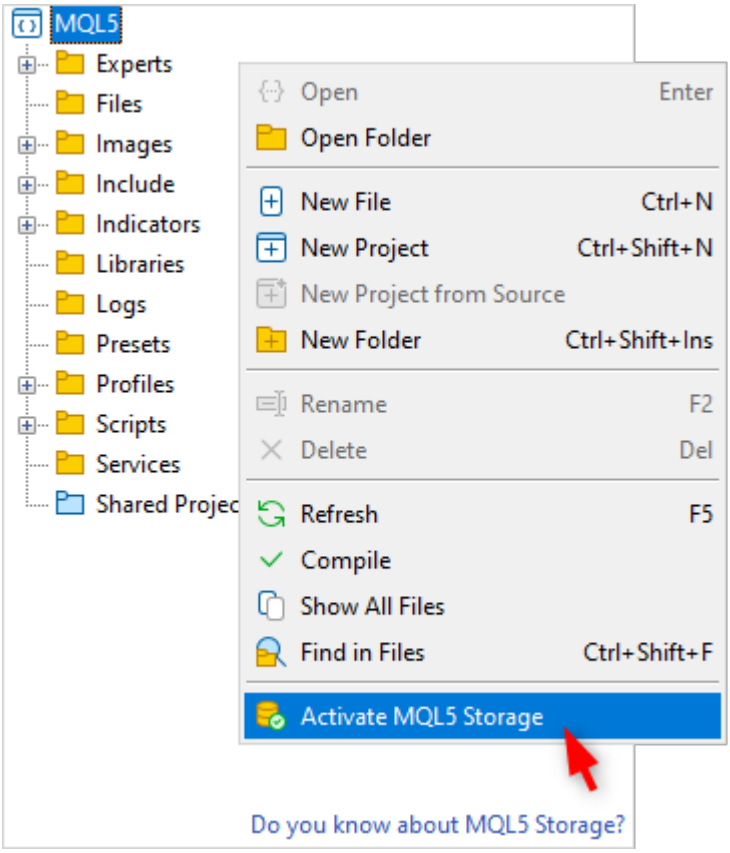
La integración de Python del lenguaje MQL5 permite ejecutar un script de Python y guardar un modelo ONNX en MetaEditor o incluso ejecutarlo en un gráfico en MetaTrader 5. Puede entrenar el modelo usando un script de Python escrito una vez con la frecuencia que necesite directamente en el terminal. Como la biblioteca contiene funciones integradas preparadas para obtener datos de precio que se pueden suministrar a la entrada del modelo ONNX:

- copy_rates_from - obtiene las barras comenzando por la fecha especificada;
- copy_rates_from_pos - obtiene las barras comenzando por el índice especificado;
- copy_ticks_range - obtiene las barras en el rango de fechas especificado;
- copy_ticks_from - obtiene los ticks comenzando por la fecha especificada;
- copy_ticks_range - obtiene los ticks en el rango de fechas especificado.

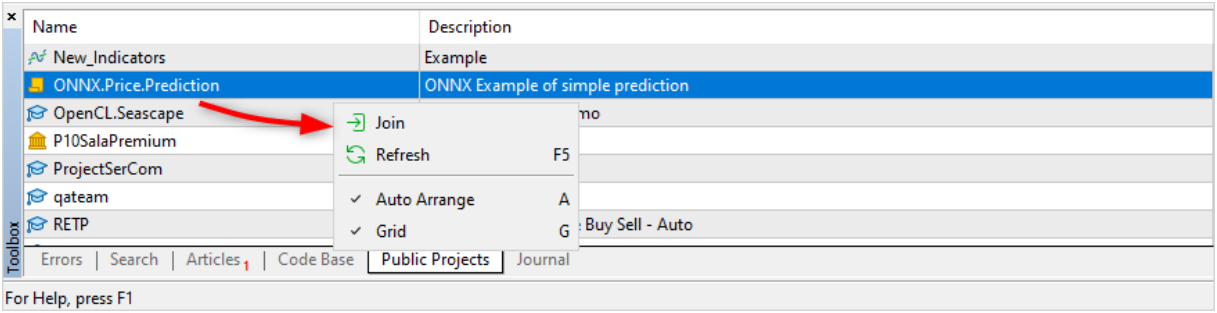
© 2000-2025, MetaQuotes Ltd.

Ejemplo de modelo preparado

Puede encontrar un ejemplo de un modelo ONNX terminado en los proyectos públicos. Para hacer esto, primero debe activar MQL5 Storage en el Navegador especificando su login MQL5 en la configuración del MetaEditor, distinguiendo entre mayúsculas y minúsculas.

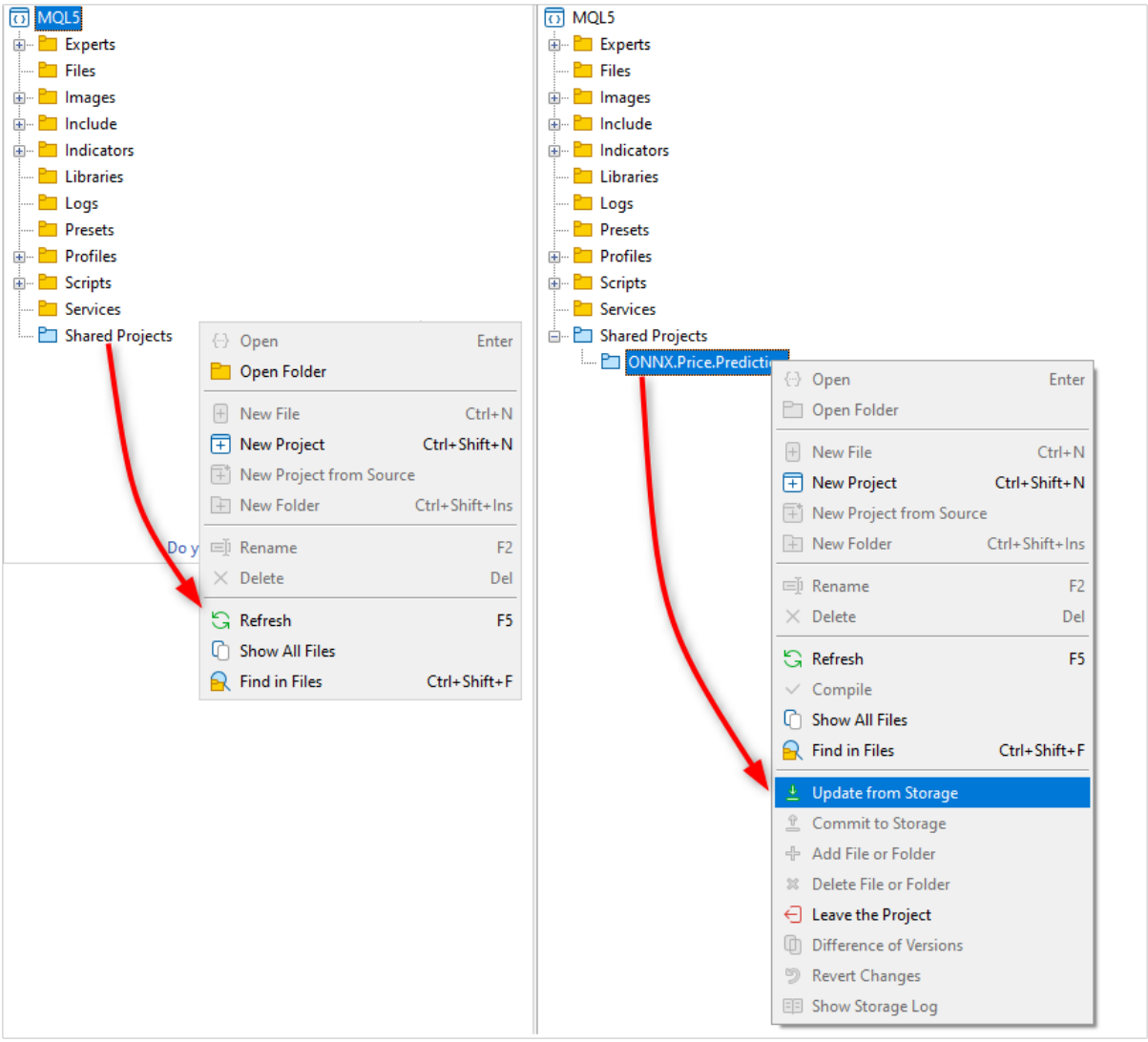


Después de la activación, busque el proyecto ONNX.Price.Prediction y únase a él usando el comando del menú contextual.



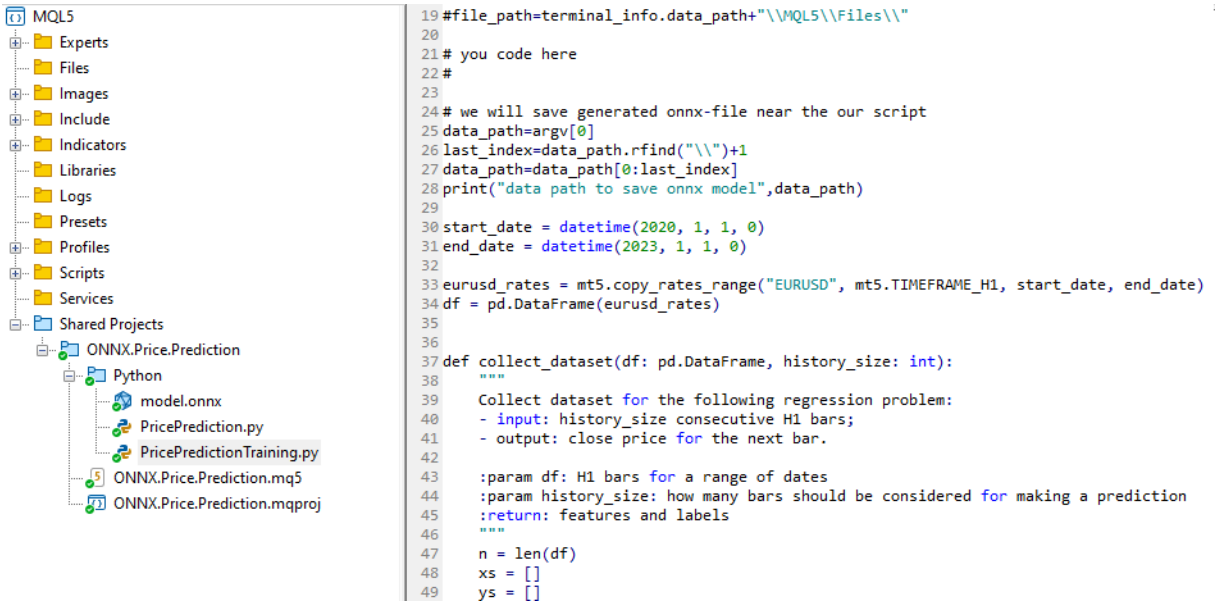
A continuación, actualice el proyecto desde MQL5 Storage.

© 2000-2025, MetaQuotes Ltd.



Después de eso, para usted estará disponible un proyecto en el que encontrará un modelo ONNX listo para usar, dos scripts de python, un script MQL5 para el funcionamiento del proyecto y un archivo de proyecto MQL5 (ONNX.Price.Prediction.mqproj).

© 2000-2025, MetaQuotes Ltd.

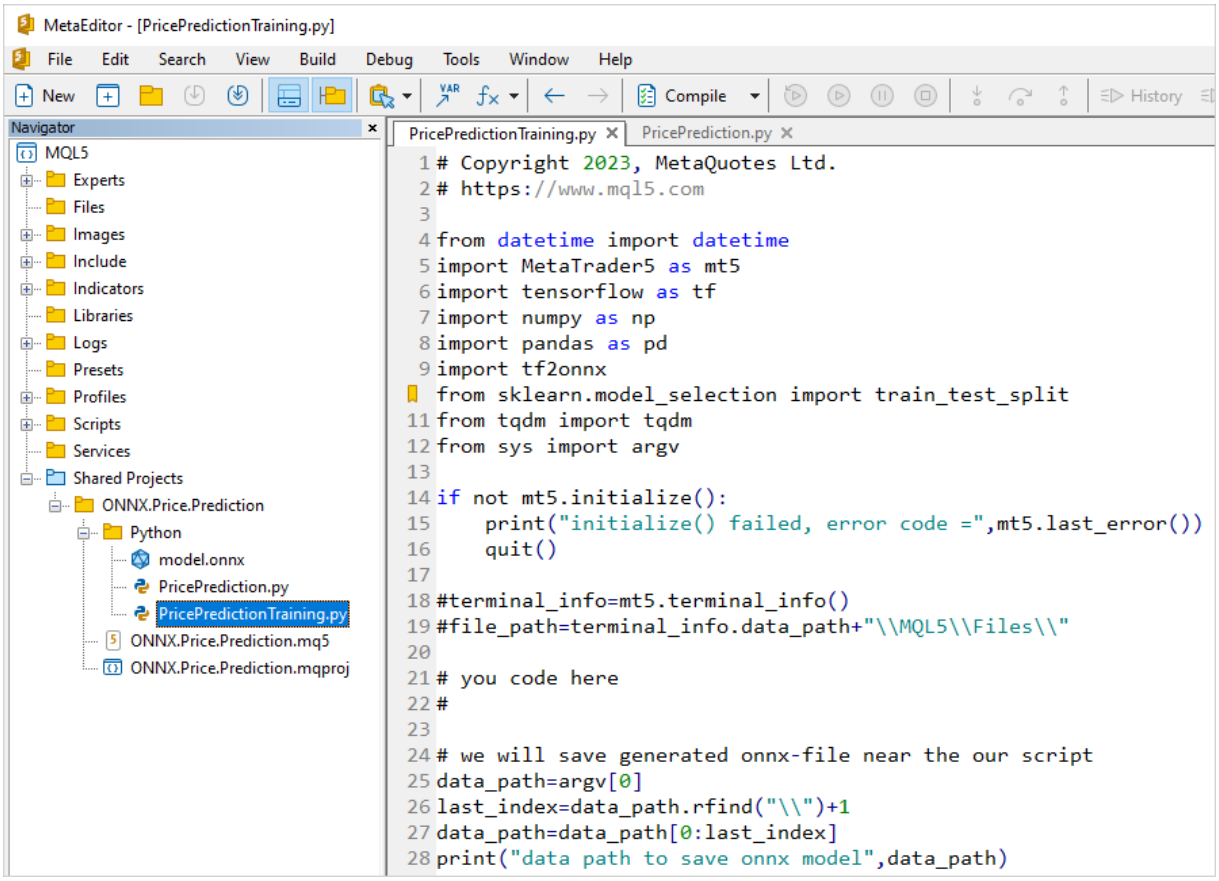


Puede crear un modelo ONNX usted mismo utilizando el script PricePredictionTraining.py incluido en el proyecto. Para hacer esto, primero debe instalar los módulos necesarios en la computadora desde la línea de comando.

```
python.exe -m pip install --upgrade pip
python -m pip install --upgrade tensorflow
python -m pip install --upgrade pandas
python -m pip install --upgrade scikit-learn
python -m pip install --upgrade matplotlib
python -m pip install --upgrade tqdm
python -m pip install --upgrade metatrader5
python -m pip install --upgrade onnx==1.12
python -m pip install --upgrade tf2onnx
python -m pip install --upgrade numpy
python -m pip install onnxruntime
```

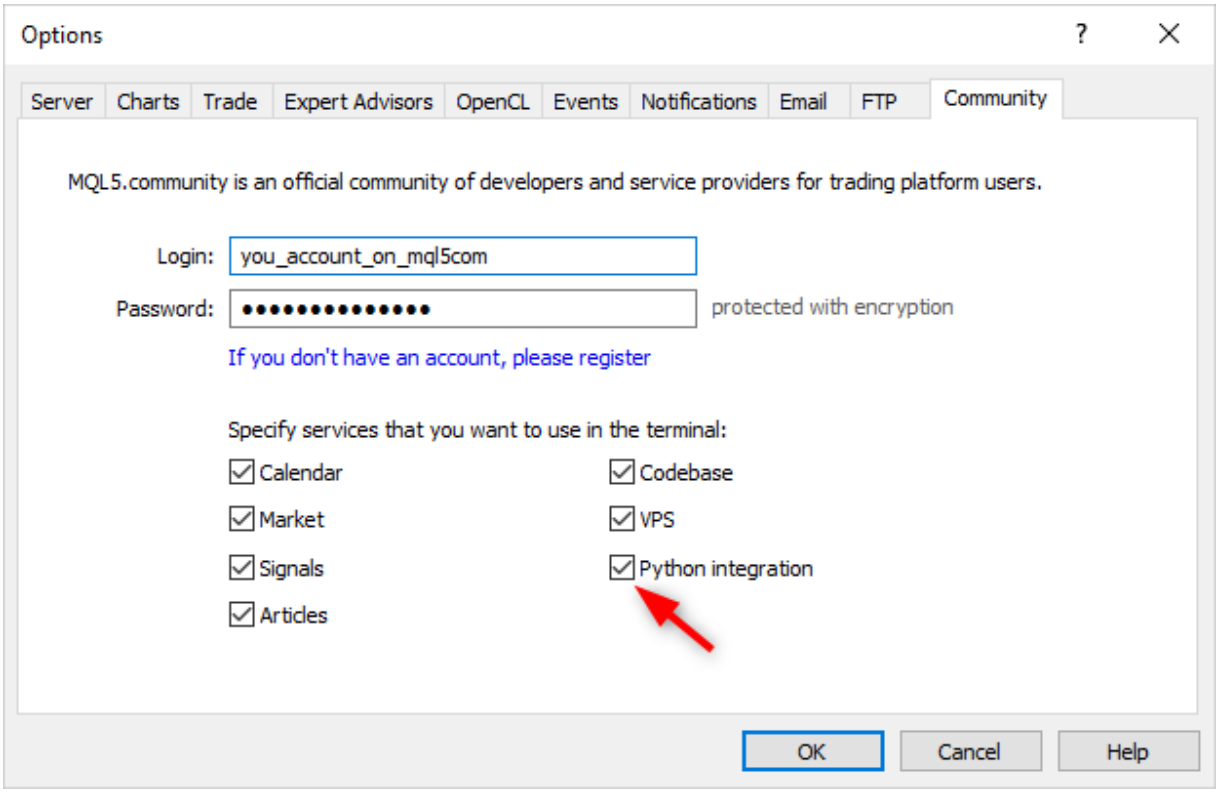
Después de instalar los módulos, abra el script PricePredictionTraining.py en MetaEditor y ejecútelo con el botón "Compilar" o con la tecla F7.

© 2000-2025, MetaQuotes Ltd.

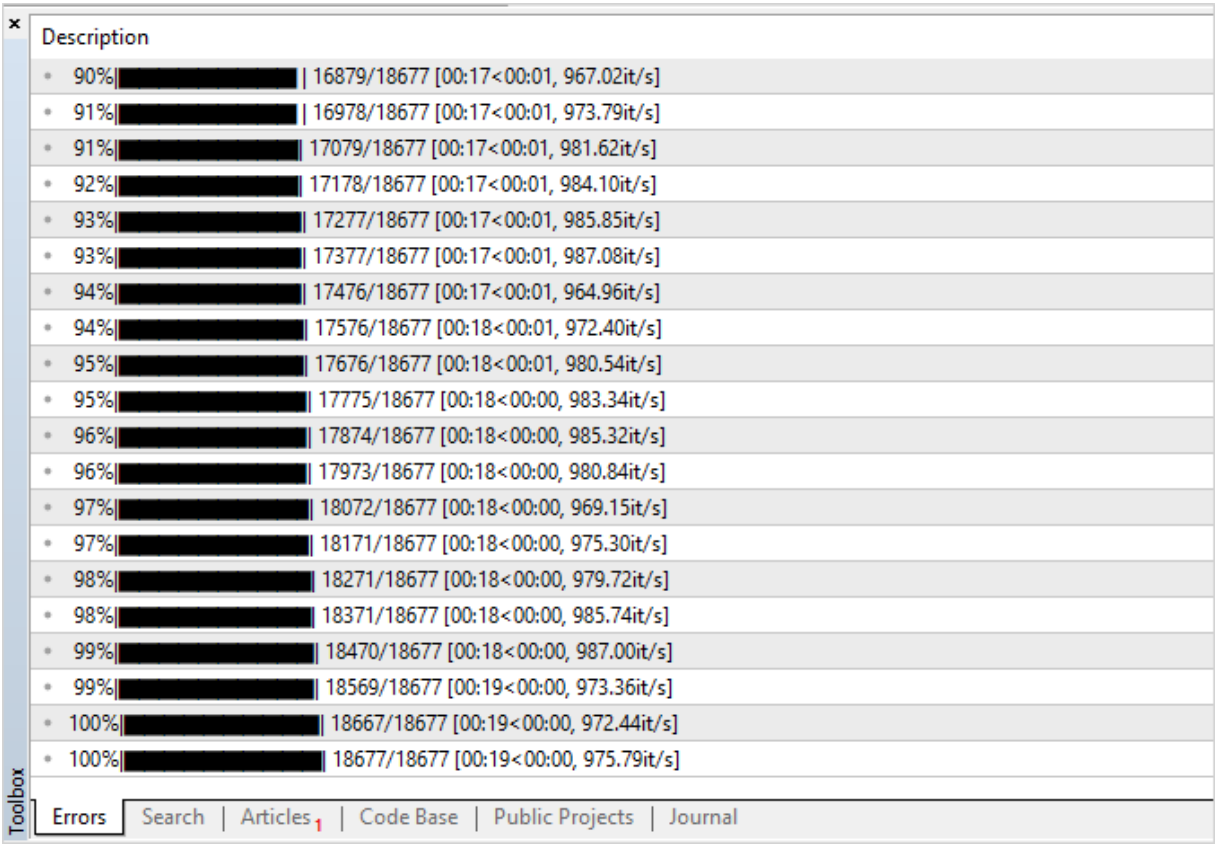


Antes de ejecutar el script Python, asegúrese de que el terminal MetaTrader 5 esté conectado a un servidor que tenga el símbolo EURUSD. Por ejemplo, conéctese al servidor MetaQuotes-Demo y compruebe en los ajustes del terminal la casilla de verificación "Integración con Python"

© 2000-2025, MetaQuotes Ltd.



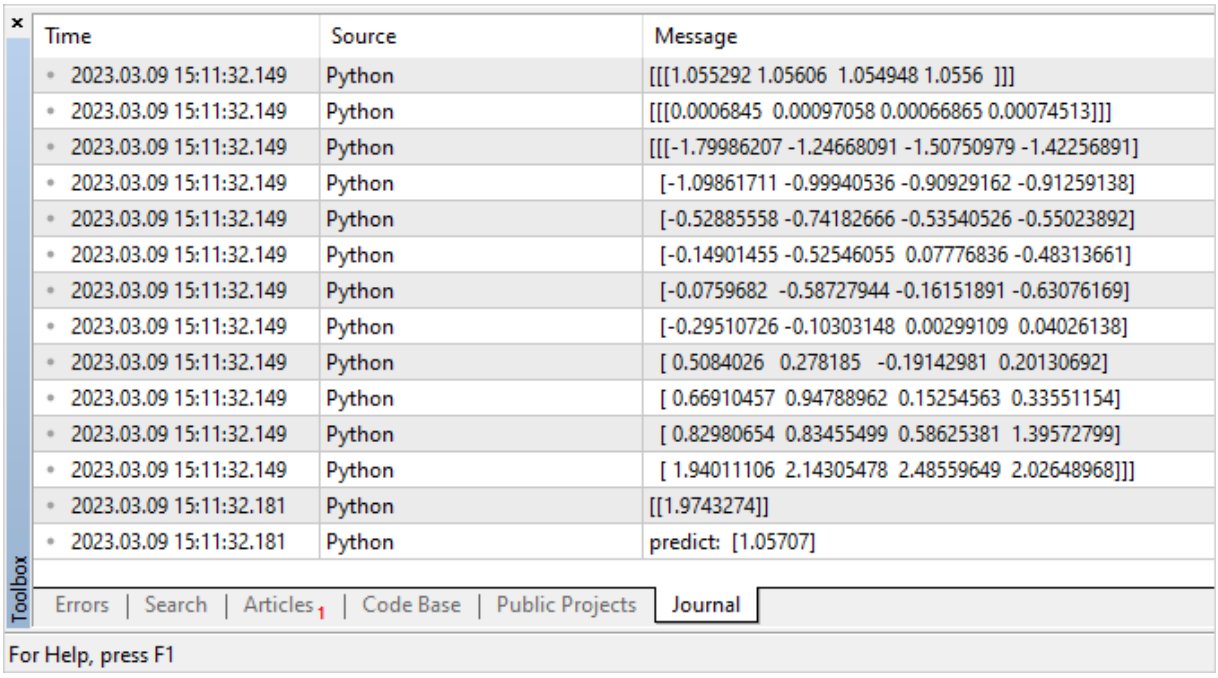
Durante el entrenamiento de la red, el MetaEditor mostrará mensajes del script de Python hasta que se complete el entrenamiento.



© 2000-2025, MetaQuotes Ltd.

Cuando el resultado es del 100%, el modelo ONNX está listo y se guarda en la carpeta del proyecto <directorio de datos del terminal>\MQL5\Shared Projects\ONNX.Price.Prediction\Python.

Puede comprobar el modelo resultante ejecutando el segundo script PricePrediction.py haciendo clic en el botón **F7**.



Time	Source	Message
2023.03.09 15:11:32.149	Python	[[[1.055292 1.05606 1.054948 1.0556]]]
2023.03.09 15:11:32.149	Python	[[[0.0006845 0.00097058 0.00066865 0.00074513]]]
2023.03.09 15:11:32.149	Python	[[[-1.79986207 -1.24668091 -1.50750979 -1.42256891]
2023.03.09 15:11:32.149	Python	[-1.09861711 -0.99940536 -0.90929162 -0.91259138]
2023.03.09 15:11:32.149	Python	[-0.52885558 -0.74182666 -0.53540526 -0.55023892]
2023.03.09 15:11:32.149	Python	[-0.14901455 -0.52546055 0.07776836 -0.48313661]
2023.03.09 15:11:32.149	Python	[-0.0759682 -0.58727944 -0.16151891 -0.63076169]
2023.03.09 15:11:32.149	Python	[-0.29510726 -0.10303148 0.00299109 0.04026138]
2023.03.09 15:11:32.149	Python	[0.5084026 0.278185 -0.19142981 0.20130692]
2023.03.09 15:11:32.149	Python	[0.66910457 0.94788962 0.15254563 0.33551154]
2023.03.09 15:11:32.149	Python	[0.82980654 0.83455499 0.58625381 1.39572799]
2023.03.09 15:11:32.149	Python	[1.94011106 2.14305478 2.48559649 2.02648968]]]
2023.03.09 15:11:32.181	Python	[[1.9743274]]
2023.03.09 15:11:32.181	Python	predict: [1.05707]

© 2000-2025, MetaQuotes Ltd.

Inicio del modelo

Para iniciar un modelo ONNX en MQL5, deberemos ejecutar 3 pasos:

1. Cargarlo desde una archivo *.onnx con ayuda de la función OnnxCreate o desde un array con ayuda de OnnxCreateFromBuffer.
2. Indicar el formulario de los datos de entrada y salida con las funciones OnnxSetInputShape y OnnxSetOutputShape.
3. Iniciar el modelo con la ayuda de OnnxRun, transmitiéndole los parámetros de entrada y salida.
4. Finalizar (en caso necesario) el funcionamiento del modelo con ayuda de OnnxRelease.

Al crear un modelo ONNX, deberemos considerar los límites y limitaciones existentes descritos en https://github.com/microsoft/onnxruntime/blob/rel-1.14.0/docs/OperatorKernels.md

Aquí tenemos algunos de ellos:

Operación	Tipos de datos soportados
ReduceSum	tensor(double), tensor(float), tensor(int32), tensor(int64)
Mul	tensor(bfloat16), tensor(double), tensor(float), tensor(float16), tensor(int32), tensor(int64), tensor(uint32), tensor(uint64)

Más abajo le mostramos un ejemplo de código MQL5 del proyecto público ONNX.Price.Prediction.

```
const long   ExtOutputShape[] = {1,1};     // formulario de datos de entrada del modelo
const long   ExtInputShape [] = {1,10,4}; // formulario de datos de salida del modelo
#resource "Python/model.onnx" as uchar ExtModel[]// modelo en forma de recurso
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- obtenemos 10 barras
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- suministramos a la entrada un conjunto de vectores OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- rellenamos las matrices de normalización
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- normalizamos los datos de entrada
   x_norm-=mm;
   x_norm/=ms;
//--- creamos el modelo
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- indicamos el formulario de los datos de entrada
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- indicamos el formulario de los datos de salida
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- convertimos los datos de entrada normalizados al tipo float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- aquí obtenemos los datos de entrada del modelo, el pronóstico del precio
   vectorf y_norm(1);
//--- iniciamos el modelo
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- mostramos en el log el valor de salida del modelo
   Print(y_norm);
//--- realizamos la conversión inversa para obtener el precio pronosticado
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
//--- finalizamos el funcionamiento
   OnnxRelease(handle);
   return(0);
  }
```

Ejemplo de ejecución del script:

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is tru
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
```

© 2000-2025, MetaQuotes Ltd.

```
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
[0.28188983]
predicted 1.0559258806393044
```

El terminal MetaTrader 5 ha seleccionado por sí mismo el ejecutor óptimo para realizar los cálculos — ONNX Runtime Execution Provider. En este caso, el modelo ha trabajado utilizando la CPU.

Vamos a cambiar el script para calcular el porcentaje de previsiones existosas del precio Close usando como base las 10 barras anteriores.

```
#resource "Python/model.onnx" as uchar ExtModel[]// modelo en forma de recurso

#define TESTS 10000  // número de muestras de prueba
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- creamos el modelo
   long session_handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
   if(session_handle==INVALID_HANDLE)
     {
      Print("Cannot create model. Error ",GetLastError());
      return(-1);
     }

//--- como para el modelo no se ha determinado el tamaño del tensor de entrada, lo est
//--- el primer índice es el tamaño del paquete; el segundo, el tamaño de la serie; el
   const long input_shape[]={1,10,4};
   if(!OnnxSetInputShape(session_handle,0,input_shape))
     {
      Print("OnnxSetInputShape error ",GetLastError());
      return(-2);
     }

//--- como para el modelo no se ha determinado el tamaño del tensor de salida, lo esta
//--- el primer índice es el tamaño del paquete, debe corresponderse con el tamaño del
//--- el segundo índice, es el número de precios pronosticados (solo pronosticamos Clo
   const long output_shape[]={1,1};
   if(!OnnxSetOutputShape(session_handle,0,output_shape))
     {
      Print("OnnxSetOutputShape error ",GetLastError());
      return(-3);
     }
//--- iniciamos las pruebas
```

© 2000-2025, MetaQuotes Ltd.

```
   vector closes(TESTS);      // vector para almacenar los precios de verificación
   vector predicts(TESTS);    // vector para almacenar las previsiones obtenidas
   vector prev_closes(TESTS); // vector para almacenar los penúltimos precios

   matrix rates;              // matriz para obtener la serie OHLC
   matrix splitted[2];        // dos submatrices para dividir la serie en serie de pru
   ulong  parts[]={10,1};     // tamaño de las submatrices divididas

//--- comenzaremos desde la barra anterior
   for(int i=1; i<=TESTS; i++)
     {
      //--- obtenemos 11 barras
      rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,i,11);
      //--- dividimos la matriz en matriz de prueba y de verificación
      rates.Vsplit(parts,splitted);
      //--- tomamos de la matriz de prueba el precio Close
      closes[i-1]=splitted[1][3][0];
      //--- último Close en la serie probada
      prev_closes[i-1]=splitted[0][3][9];

      //--- enviamos a la prueba una matriz de prueba de 10 barras
      predicts[i-1]=PricePredictionTest(session_handle,splitted[0]);
      //--- error de ejecución
      if(predicts[i-1]<=0)
        {
         OnnxRelease(session_handle);
         return(-4);
        }
     }
//--- finalizamos el funcionamiento
   OnnxRelease(session_handle);
//--- evaluamos la corrección del movimiento de precio pronosticado
   int    right_directions=0;
   vector delta_predicts=prev_closes-predicts;
   vector delta_actuals=prev_closes-closes;

   for(int i=0; i<TESTS; i++)
      if((delta_predicts[i]>0 && delta_actuals[i]>0) || (delta_predicts[i]<0 && delta_
         right_directions++;
   PrintFormat("right direction predictions = %.2f%%",(right_directions*100.0)/double
//---
   return(0);
  }
//+------------------------------------------------------------------+
//|  Preparamos los datos e iniciamos el modelo                      |
//+------------------------------------------------------------------+
double PricePredictionTest(const long session_handle,matrix& rates)
  {
   static matrixf input_data(10,4); // matriz para convertir los datos de entrada
```

© 2000-2025, MetaQuotes Ltd.

```
   static vectorf output_data(1);    // vector para obtener el resultado
   static matrix mm(10,4);           // matriz de vectores Mean horizontales
   static matrix ms(10,4);           // matriz de vectores Std horizontales

//--- debemos suministar a la entrada del modelo un conjunto de vectores OHLC vertical
   matrix x_norm=rates.Transpose();
//--- normalizamos los precios
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
   x_norm-=mm;
   x_norm/=ms;

//--- iniciamos el modelo
   input_data.Assign(x_norm);
   if(!OnnxRun(session_handle,ONNX_DEBUG_LOGS,input_data,output_data))
     {
      Print("OnnxRun error ",GetLastError());
      return(0);
     }
//--- renormalizamos el precio del valor de salida
   double y_pred=output_data[0]*s[3]+m[3];

   return(y_pred);
  }
```

Ejecutamos el script y obtenemos una precisión de predicción de alrededor del 51%

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is tru
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
right direction predictions = 51.34 %
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Validación de modelos en el simulador de estrategias

Los modelos para trabajar en los mercados financieros se pueden probar en el simulador de estrategias del terminal MetaTrader 5. Esta es la opción más rápida y cómoda, pues no requiere esfuerzos adicionales para emular el entorno del mercado y las condiciones comerciales.

Para probar el modelo, reescribimos el código del proyecto público ONNX.Price.Prediction en un asesor experto. Esto requerirá ediciones menores.

Vamos a trasladar la creación del modelo a la función OnInit, y la finalización de la sesión onnx a OnDeinit. El bloque principal para trabajar con el modelo se colocará en el controlador OnTick.

Añadiremos también la obtención del precio de cierre de las 2 barras anteriores para comparar el precio de cierre actual y la predicción.

El código del asesor es pequeño y fácil de leer.

```
const long   ExtInputShape [] = {1,10,4}; // formulario de datos de salida del modelo
const long   ExtOutputShape[] = {1,1};    // formulario de datos de entrada del modelo
#resource "Python/model.onnx" as uchar ExtModel[];// modelo en forma de recurso

long handle;          // manejador del modelo
ulong predictions=0; // contador de pronósticos
ulong confirmed=0;   // contador de pronósticos exitosos
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- comprobaciones básicas
   if(_Symbol!="EURUSD")
     {
      Print("Symbol must be EURUSD, testing aborted");
      return(-1);
     }
   if(_Period!=PERIOD_H1)
     {
      Print("Timeframe must be H1, testing aborted");
      return(-1);
     }
//--- creamos el modelo
   handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- indicamos el formulario de los datos de entrada
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

© 2000-2025, MetaQuotes Ltd.

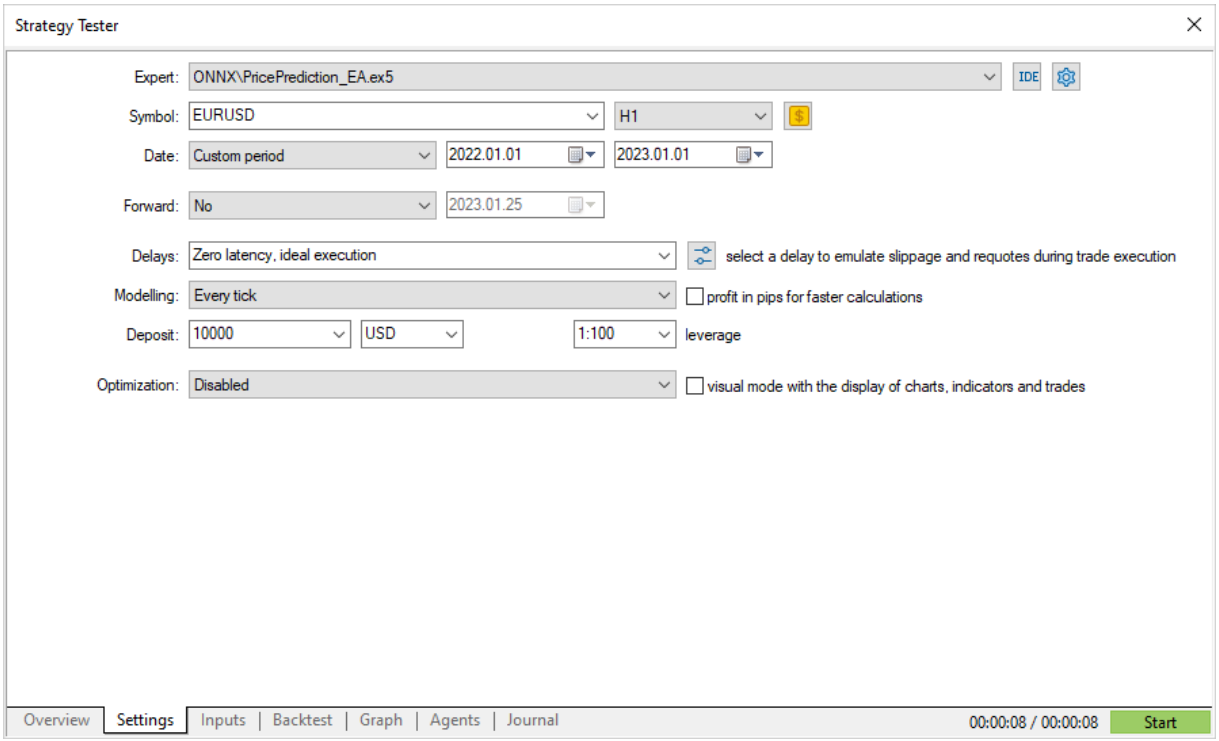
```
//--- indicamos el formulario de los datos de salida
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- hemos finalizado el funcionamiento del modelo
   OnnxRelease(handle);
//--- calculamos y mostramos las estadísticas de los pronósticos
   PrintFormat("Successfull predictions = %.2f %%",confirmed*100./double(predictions))
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   static datetime open_time=0;
   static double predict;
//--- comprobamos la hora de apertura de la barra actual
   datetime time=iTime(_Symbol,_Period,0);
   if(time==0)
     {
      PrintFormat("Failed to get Time(0), error %d", GetLastError());
      return;
     }
//--- si la hora de apertura de la barra no ha cambiado, salimos hasta la siguiente li
   if(time==open_time)
      return;
//--- obtenemos los precios de cierre de las 2 últimas barras finalizadas
   double close[];
   int recieved=CopyClose(_Symbol,_Period,1,2,close);
   if(recieved!=2)
     {
      PrintFormat("CopyClose(2 bars) failed, error %d",GetLastError());
      return;
     }
   double delta_predict=predict-close[0]; // cambio de precio pronosticado
   double delta_actual=close[1]-close[0]; // cambio de precio real
   if((delta_predict>0 && delta_actual>0) || (delta_predict<0 && delta_actual<0))
      confirmed++;
```

© 2000-2025, MetaQuotes Ltd.

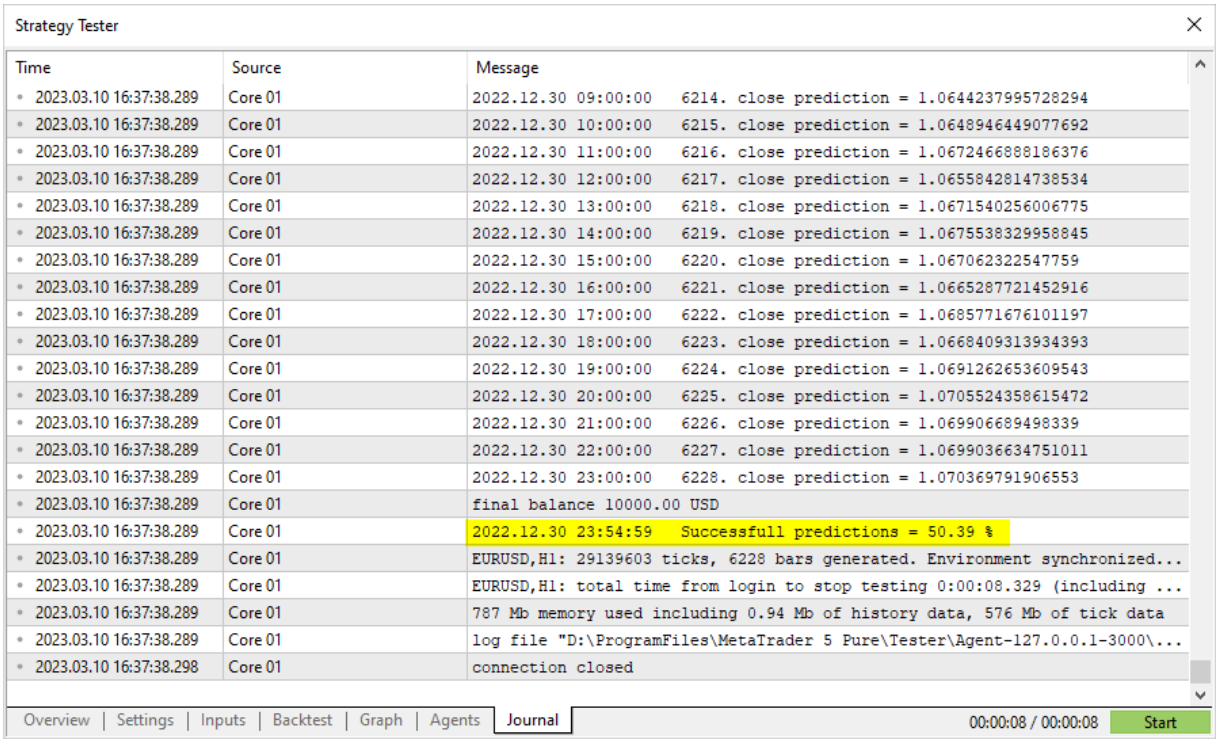
```
//--- calculamos el precio de cierre de la nueva barra para comprobarlo en la siguient
   matrix rates;
//--- obtenemos 10 barras
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,1,10))
      return;
//--- suministramos a la entrada un conjunto de vectores OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- rellenamos las matrices de normalización
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- normalizamos los datos de entrada
   x_norm-=mm;
   x_norm/=ms;
//--- convertimos los datos de entrada normalizados al tipo float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- aquí obtenemos los datos de entrada del modelo, el pronóstico del precio
   vectorf y_norm(1);
//--- iniciamos el modelo
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
     }
//--- hacemos la transformación inversa para obtener el precio pronosticado y comproba
   predict=y_norm[0]*s[3]+m[3];
   predictions++;  // incrementamos el contador de pronósticos
   Print(predictions,". close prediction = ",predict);
//--- recordamos la hora de apertura de la barra para comprobar el próximo tick
   open_time=time;
  }
```

Compilamos el asesor y comenzamos la prueba en el intervalo de 2022, indicamos el símbolo EURUSD y el marco temporal H1, en los que se entrenó el modelo. El modo de simulación de ticks no importa, porque el código contiene la comprobación de la aparición de una nueva barra.

© 2000-2025, MetaQuotes Ltd.



Iniciamos y obtenemos el resultado en el diario de registro de la prueba: algo más del 50% de pronósticos correctos en 2022.



Si los resultados de la verificación preliminar del modelo son satisfactorios, podemos comenzar a escribir una estrategia comercial completa usándolos.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

OnnxCreate

Crea una sesión de ONNX cargando el modelo desde un archivo *.onnx.

```
long  OnnxCreate(
   string  filename,  // ruta al archivo
   uint    flags      // banderas para crear el modelo
   );
```

Parámetros

filename

[in] Ruta al archivo *.onnx del modelo respecto a la carpeta \MQL5\Files\.

flags

[in] Las banderas ENUM_ONNX_FLAGS que describen el modo de creación del modelo son ONNX_COMMON_FOLDER y ONNX_DEBUG_LOGS.

Valor retornado

Manejador para la sesión creada o INVALID_HANDLE en el caso de error. Para obtener el código de error, llame a la función GetLastError.

Observación

Si el archivo especificado no se encuentra en el disco, se hará un nuevo intento de abrir el archivo añadiendo al nombre la extensión '.onnx'.

© 2000-2025, MetaQuotes Ltd.

OnnxCreateFromBuffer

Crea una sesión de ONNX cargando el modelo desde un array de datos.

```
long  OnnxCreateFromBuffer(
   const uchar&  buffer[],   // enlace al array
   ulong         flags       // banderas para crear el modelo
   );
```

Parámetros

buffer

[in] Array con los datos ONNX del modelo.

flags

[in] Las banderas ENUM_ONNX_FLAGS que describen el modo de creación del modelo son ONNX_COMMON_FOLDER y ONNX_DEBUG_LOGS.

Valor retornado

Manejador para la sesión creada o INVALID_HANDLE en el caso de error. Para obtener el código de error, llame a la función GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxRelease

Eliminar sesión ONNX.

```
bool  OnnxRelease(
   long   onnx_handle  // manejador de la sesión ONNX
   );
```

Parámetros

onnx_handle

[in] Manejador del objeto de sesión ONNX creado a través de OnnxCreate o OnnxCreateFromBuffer.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código de error, llame a la función GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxRun

Iniciar un modelo ONNX para su ejecución.

```
bool  OnnxRun(
   long    onnx_handle,  // manejador de la sesión ONNX
   ulong   flags,        // banderas que describen el modo de inicio
   ...                   // parámetros de entrada y salida del modelo
   );
```

Parámetros

onnx_handle

[in] Manejador del objeto de sesión ONNX creado a través de OnnxCreate o OnnxCreateFromBuffer.

flags

[in] Las banderas ENUM_ONNX_FLAGS que describen el modo de inicio son ONNX_DEBUG_LOGS y ONNX_NO_CONVERSION.

...

[in] [out] Parámetros de entrada y salida del modelo.

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código de error, llame a la función GetLastError.

ENUM_ONNX_FLAGS

Identificador	Descripción
ONNX_DEBUG_LOGS	Muestra de logs de depuración
ONNX_NO_CONVERSION	Prohibición de conversión automática, use los datos del usuario tal cual
ONNX_COMMON_FOLDER	Carga del archivo del modelo desde la carpeta Common\Files, el valor es igual al indicador FILE_COMMON

Ejemplo:

```
const long                            ExtOutputShape[] = {1,1};     // formulario de c
const long                            ExtInputShape [] = {1,10,4}; // formulario de c
#resource "Python/model.onnx" as uchar ExtModel[]                   // modelo en forma
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- obtenemos 10 barras
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- suministramos a la entrada un conjunto de vectores OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- rellenamos las matrices de normalización
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- normalizamos los datos de entrada
   x_norm-=mm;
   x_norm/=ms;
//--- creamos el modelo
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- indicamos el formulario de los datos de entrada
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- indicamos el formulario de los datos de salida
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- convertimos los datos de entrada normalizados al tipo float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- aquí obtenemos los datos de entrada del modelo, el pronóstico del precio
   vectorf y_norm(1);
//--- iniciamos el modelo
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- mostramos en el log el valor de salida del modelo
   Print(y_norm);
//--- realizamos la conversión inversa para obtener el precio pronosticado
```

© 2000-2025, MetaQuotes Ltd.

```
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
//--- finalizamos el funcionamiento
   OnnxRelease(handle);
   return(0);
  };
```

Ver también

OnnxSetInputShape, OnnxSetOutputShape

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputCount

Obtener el número de parámetros de entrada del modelo ONNX.

```
long  OnnxGetInputCount(
   long   onnx_handle  // manejador de la sesión ONNX
   );
```

Parámetros

onnx_handle

[in] Manejador del objeto de sesión ONNX creado a través de OnnxCreate o OnnxCreateFromBuffer.

Valor retornado

Si la ejecución tiene éxito, retornará el número de parámetros de entrada; de lo contrario, -1. Para obtener el código de error, llame a la función GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputCount

Obtener el número de parámetros de salida del modelo ONNX.

```
long  OnnxGetOutputCount(
   long   onnx_handle  // manejador de la sesión ONNX
   );
```

Parámetros

onnx_handle

[in] Manejador del objeto de sesión ONNX creado a través de OnnxCreate o OnnxCreateFromBuffer.

Valor retornado

Si la ejecución tiene éxito, retornará el número de parámetros de salida; de lo contrario, -1. Para obtener el código de error, llame a la función GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputName

Obtener el nombre del parámetro de entrada del modelo según el índice.

```
string  OnnxGetInputName(
   long   onnx_handle,  // manejador de la sesión ONNX
   long   index         // índice del parámetro
   );
```

Parámetros

onnx_handle

[in] Manejador del objeto de sesión ONNX creado a través de OnnxCreate o OnnxCreateFromBuffer.

index

[in] Índice del parámetro de entrada, comenzando desde 0.

Valor retornado

Si la ejecución tiene éxito, retornará el nombre del parámetro de entrada; de lo contrario, NULL. Para obtener el código de error, llame a la función GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputName

Obtener el nombre de un parámetro de salida del modelo según el índice.

```
string  OnnxGetOutputName(
   long   onnx_handle,  // manejador de la sesión ONNX
   long   index         // índice del parámetro
   );
```

Parámetros

onnx_handle

[in] Manejador del objeto de sesión ONNX creado a través de OnnxCreate o OnnxCreateFromBuffer.

index

[in] Índice del parámetro de salida, comenzando desde 0.

Valor retornado

Si la ejecución tiene éxito, retornará el nombre del parámetro de salida; de lo contrario, NULL. Para obtener el código de error, llame a la función GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputTypeInfo

Obtener la descripción del tipo de parámetro de entrada del modelo.

```
bool  OnnxGetInputTypeInfo(
   long           onnx_handle,  // manejador de la sesión ONNX
   long           index,        // índice del parámetro
   OnnxTypeInfo&  typeinfo      // descripción del tipo de parámetro
   );
```

Parámetros

onnx_handle

[in] Manejador del objeto de sesión ONNX creado a través de OnnxCreate o OnnxCreateFromBuffer.

index

[in] Índice del parámetro de entrada, comenzando desde 0.

typeinfo

[out] Estructura OnnxTypeInfo que describe el tipo de parámetro de entrada.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código de error, llame a la función GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputTypeInfo

Obtener la descripción del tipo de parámetro de salida del modelo.

```
bool  OnnxGetOutputTypeInfo(
   long           onnx_handle,  // manejador de la sesión ONNX
   long           index,        // índice del parámetro
   OnnxTypeInfo&  typeinfo      // descripción del tipo de parámetro
   );
```

Parámetros

onnx_handle

[in] Manejador del objeto de sesión ONNX creado a través de OnnxCreate o OnnxCreateFromBuffer.

index

[in] Índice del parámetro de salida, comenzando desde 0.

typeinfo

[out] Estructura OnnxTypeInfo que describe el tipo de parámetro de salida.

Valor retornado

Si se ejecuta con éxito, retorna true, de lo contrario, false. Para obtener el código de error, llame a la función GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxSetInputShape

Establece la dimensionalidad de los datos de entrada del modelo según el índice.

```
bool  OnnxSetInputShape(
   long           onnx_handle,  // manejador de la sesión ONNX
   long           input_index,  // índice del parámetro de entrada
   const ulong&   shape[]       // array que describe la dimensionalidad de los datos d
   );
```

Parámetros

onnx_handle

[in] Manejador del objeto de sesión ONNX creado a través de OnnxCreate o OnnxCreateFromBuffer.

input_index

[in] Índice del parámetro de entrada, comenzando desde 0.

shape

[in] Matriz que describe la dimensionalidad de los datos de entrada del modelo.

Valor retornado

Si la ejecución tiene éxito, retornará el nombre del parámetro de entrada; de lo contrario, NULL. Para obtener el código de error, llame a la función GetLastError.

Ejemplo:

```
//---- describimos la dimensionalidad de los datos de entrada y salida del modelo
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- creamos el modelo
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- indicamos el formulario de los datos de entrada
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- indicamos el formulario de los datos de salida
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Ver también

© 2000-2025, MetaQuotes Ltd.

OnnxSetOutputShape

© 2000-2025, MetaQuotes Ltd.

OnnxSetOutputShape

Establece la dimensionalidad de los datos de salida del modelo según el índice.

```
bool  OnnxSetOutputShape(
   long           onnx_handle,    // manejador de la sesión ONNX
   long           output_index,   // índice del parámetro de salida
   const ulong&   shape[]         // array que describe la dimensionalidad de los datos
   );
```

Parámetros

onnx_handle

[in] Manejador del objeto de sesión ONNX creado a través de OnnxCreate o OnnxCreateFromBuffer.

output_index

[in] Índice del parámetro de salida, comenzando desde 0.

shape

[in] Matriz que describe la dimensionalidad de los datos de salida del modelo.

Valor retornado

Si la ejecución tiene éxito, retornará el nombre del parámetro de entrada; de lo contrario, NULL. Para obtener el código de error, llame a la función GetLastError.

Ejemplo:

```
//---- describimos la dimensionalidad de los datos de entrada y salida del modelo
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- creamos el modelo
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- indicamos el formulario de los datos de entrada
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- indicamos el formulario de los datos de salida
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Ver también

© 2000-2025, MetaQuotes Ltd.

OnnxSetInputShape

© 2000-2025, MetaQuotes Ltd.

Estructura de los datos

Para trabajar con modelos ONNX se usan las siguientes estructuras de datos:

OnnxTypeInfo

La estructura describe el tipo del parámetro de entrada o salida del modelo ONNX

```
struct OnnxTypeInfo
  {
   ENUM_ONNX_TYPE          type;           // tipo de parámetro
   OnnxTensorTypeInfo      tensor;         // descripción del tensor
   OnnxMapTypeInfo         map;            // descripción de map
   OnnxSequenceTypeInfo    sequence;       // descripción de la secuencia
  };
```

Solo el tensor (ONNX_TYPE_TENSOR) puede usarse como parámetro de entrada, en cuyo caso solo el campo OnnxTypeInfo::tensor se rellenará con valores, los otros campos (map y sequence) serán indefinidos.

Solo uno de los tres tipos de OnnxTypeInfo (ONNX_TYPE_TENSOR, ONNX_TYPE_MAP o ONNX_TYPE_SEQUENCE) se puede utilizar como parámetro de salida, dependiendo del tipo se rellenará la subestructura correspondiente (OnnxTypeInfo::tensor, OnnxTypeInfo::map u OnnxTypeInfo::sequence).

OnnxTensorTypeInfo

La estructura describe el tensor en el parámetro de entrada o salida del modelo ONNX

```
struct OnnxTensorTypeInfo
  {
   const ENUM_ONNX_DATA_TYPE    data_type;      // tipo de datos en el tensor
   const long                   dimensions[];   // número de elementos en el tensor
  };
```

OnnxMapTypeInfo

La estructura describe map, que se obtiene en el parámetro de salida del modelo ONNX

```
struct OnnxMapTypeInfo
  {
   const ENUM_ONNX_DATA_TYPE    key_type;       // tipo de clave
   const OnnxTypeInfo&          value_type;     // tipo de valor
  };
```

OnnxSequenceTypeInfo

La estructura describe la sencuencia, que se obtiene en el parámetro de salida del modelo ONNX

© 2000-2025, MetaQuotes Ltd.

```
struct OnnxSequenceTypeInfo
  {
   const OnnxTypeInfo&           value_type;      // tipo de datos en la secuencia
  };
```

ENUM_ONNX_TYPE

La enumeración **ENUM_ONNX_TYPE** describe el tipo del parámetro del modelo

Identificador	Descripción
ONNX_TYPE_UNKNOWN	Desconocido
ONNX_TYPE_TENSOR	Tensor
ONNX_TYPE_SEQUENCE	Secuencia
ONNX_TYPE_MAP	Mapa
ONNX_TYPE_OPAQUE	Abstracto
ONNX_TYPE_SPARSETENSOR	Tensor disperso

ENUM_ONNX_DATA_TYPE

La enumeración **ENUM_ONNX_DATA_TYPE** describe el tipo de datos utilizados

Identificador	Descripción
ONNX_DATA_TYPE_UNDEFINED	No definido
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_INT8	int de 8 bits
ONNX_DATA_TYPE_UINT16	uint de 16 bits
ONNX_DATA_TYPE_INT16	int de 16 bits
ONNX_DATA_TYPE_INT32	int de 32 bits
ONNX_DATA_TYPE_INT64	int de 64 bits
ONNX_DATA_TYPE_STRING	string
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT16	float de 16 bits
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	uint de 32 bits
ONNX_DATA_TYPE_UINT64	uint de 64 bits
ONNX_DATA_TYPE_COMPLEX64	número complejo de 64 bits

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
ONNX_DATA_TYPE_COMPLEX128	número complejo de 128 bits
ONNX_DATA_TYPE_BFLOAT16	bfloat de 16 bits (Brain Floating Point)

ENUM_ONNX_FLAGS

La enumeración **ENUM_ONNX_FLAGS** describe el modo de inicio del modelo

Identificador	Descripción
ONNX_DEBUG_LOGS	Muestra de logs de depuración
ONNX_NO_CONVERSION	Prohibición de conversión automática, use los datos del usuario tal cual
ONNX_COMMON_FOLDER	Carga del archivo del modelo desde la carpeta Common\Files, el valor es igual al indicador FILE_COMMON

Conversión de arrays al trabajar con modelos ONNX

En las tareas de aprendizaje automático, no siempre se necesita una alta precisión en los cálculos. Para acelerar los cálculos, algunos modelos utilizan tipos de datos con menor precisión, como Float16 e incluso Float8. Para permitir a los usuarios introducir dichos datos en la entrada de los modelos, MQL5 ha añadido 4 funciones especiales que convierten los tipos estándar de MQL5 en tipos especiales FP16 y FP8.

Función	Acción
ArrayToFP16	Permite copiar un array de tipo float o double en un array de tipo ushort con el formato establecido
ArrayToFP8	Permite copiar un array de tipo float o double en un array de tipo uchar con el formato establecido
ArrayFromFP16	Permite copiar un array de tipo ushort en un array de tipo float o double con el formato establecido
ArrayFromFP8	Permite copiar un array de tipo uchar en un array de tipo float o double con el formato establecido

Estas funciones utilizan los formatos especiales definidos en las enumeraciones siguientes para convertir arrays.

ENUM_FLOAT16_FORMAT

La enumeración **ENUM_FLOAT16_FORMAT** describe 2 formatos del tipo FP16.

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
FLOAT_FP16	Formato estándar de 16 bits, también conocido como half
FLOAT_BFP16	Formato especial brain float point

Ambos formatos tienen sus ventajas y sus limitaciones. FLOAT16 ofrece una mayor precisión, pero requiere más recursos de almacenamiento y cálculo. BFLOAT16, por su parte, ofrece un mayor rendimiento y eficiencia en el procesamiento de datos, pero puede ser menos preciso.

ENUM_FLOAT8_FORMAT

La enumeración **ENUM_FLOAT8_FORMAT** describe 4 formatos del tipo FP8.

El formato FP8 (coma flotante de 8 bits) es uno de los tipos de datos usados para representar números en coma flotante. En FP8, cada número se representa con 8 bits de datos, que suelen dividirse en tres componentes: signo, exponente y mantisa. Este formato ofrece un compromiso entre precisión y eficiencia de almacenamiento, lo cual lo hace atractivo para su uso en aplicaciones en las que es necesario ahorrar memoria y recursos computacionales.

Identificador	Descripción
FLOAT_FP8_E4M3FN	Un número de coma flotante de 8 bits, 4 bits de orden y 3 bits de mantisa. Se utiliza normalmente como coeficientes.
FLOAT_FP8_E4M3FNUZ	Un número de coma flotante de 8 bits, 4 bits de orden y 3 bits de mantisa. Soporta NaN, cero negativo e Inf no están soportados. Se utiliza normalmente como coeficientes.
FLOAT_FP8_E5M2FN	Un número de coma flotante de 8 bits, 5 bits de orden y 2 bits de mantisa. Soporta NaN y Inf. Normalmente se usa para gradientes.
FLOAT_FP8_E5M2FNUZ	Un número de coma flotante de 8 bits, 5 bits de orden y 2 bits de mantisa. Soporta NaN e Inf, no soporta cero negativo. También se utiliza para gradientes.

Una de las principales ventajas de FP8 es su eficiencia en el procesamiento de grandes cantidades de datos. Con su representación compacta de los números, el FP8 puede reducir los requisitos de memoria y acelerar los cálculos. Esto es especialmente importante en las aplicaciones de aprendizaje automático e inteligencia artificial, donde el procesamiento de grandes conjuntos de datos es habitual.

© 2000-2025, MetaQuotes Ltd.

Librería Estándar

Esta sección contiene detalles técnicos de la Librería Estándar de MQL5, así como las descripciones de sus componentes clave.

La Librería Estándar de MQL5 está escrita en el lenguaje MQL5, y se ha diseñado para facilitar la escritura de programas (indicadores, guiones y asesores expertos) a los usuarios finales. La Librería proporciona acceso a la mayoría de funciones internas de MQL5.

La Librería Estándar de MQL5 se encuentra en el directorio de trabajo del terminal 'Include'.

Sección	Ubicación
Matemáticas	Include\Math\
OpenCL	Include\OpenCL\
Clase base CObject	Include\
Colecciones de datos	Include\Arrays\
Colecciones de datos genéricas	Include\Generic\
Archivos	Include\Files\
Cadenas de caracteres	Include\Strings\
Objetos gráficos	Include\Objects\
Gráficos personalizados	Include\Canvas\
Gráficos 3D	Include\Canvas\
Gráficos de precios	Include\Charts\
Gráficos científicos	Include\Graphics\
Indicadores	Include\Indicators\
Clases de comercio	Include\Trade\
Módulos de estrategias	Include\Expert\
Paneles y ventanas de diálogo	Include\Controls\

© 2000-2025, MetaQuotes Ltd.

Matemáticas

Para realizar cálculos en las diferentes ramas de las matemáticas, se proponen diversas bibliotecas:

- Estadística – funciones para trabajar con distintas distribuciones de la teoría de probabilidad
- Lógica difusa – biblioteca de lógica difusa, en la que están implementados los sistemas de deducción de lógica difusa Mamdani y Sugeno
- ALGLIB – análisis de datos (agrupamiento, árbol de decisión, regresión lineal, red neuronal), solución de ecuaciones diferenciales, transformada de Fourier, integración numérica, tareas de optimización, análisis estadístico y mucho más.

© 2000-2025, MetaQuotes Ltd.

Estadística

La biblioteca estadística ha sido creada para trabajar de forma cómoda con las principales distribuciones estadísticas.

Para cada una de las distribuciones, en la biblioteca se presentan 5 funciones:

1. Cálculo de la densidad de la distribución – función del tipo MathProbabilityDensityX()
2. Cálculo de la probabilidad – función del tipo MathCumulativeDistributionX()
3. Cálculo de los cuantiles de las distribuciones – función del tipo MathQuantileX()
4. Generación de números aleatorios con la distribución indicada – función del tipo MathRandomX()
5. Cálculo de los momentos teóricos de las distribuciones – función del tipo MathMomentsX()

Aparte del cálculo de magnitudes aleatorias, en la biblioteca se dispone también de la posibilidad de cargar de nuevo funciones, que realizarán los mismos cálculos en las matrices.

- Características estadísticas
- Distribución normal
- Distribución log-normal
- Distribución beta
- Distribución beta no central
- Distribución gamma
- Distribución chi-cuadrado
- Distribución chi-cuadrado no central
- Distribución exponencial
- Distribución F
- Distribución F no central
- Distribución T
- Distribución T no central
- Distribución logística
- Distribución de Cauchy
- Distribución uniforme
- Distribución de Weibull
- Distribución binomial
- Distribución binomial negativa
- Distribución geométrica
- Distribución hipergeométrica
- Distribución de Poisson
- Funciones auxiliares

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                     NormalDistributionExample.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- activamos las funciones para calcular una distribución normal
#include <Math\Stat\Normal.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- establecemos los parámetros de la distribución normal
   double mu=5.0;
   double sigma=1.0;
   PrintFormat("Distribución normal con los parámetros mu=%G y sigma=%G, ejemplos de c
//--- establecemos el intervalo
   double x1=mu-sigma;
   double x2=mu+sigma;
//--- variables para el cálculo de la probabilidad
   double cdf1,cdf2,probability;
//--- variables para el código de error
   int error_code1,error_code2;
//--- calculamos los valores de la función de distribución
   cdf1=MathCumulativeDistributionNormal(x1,mu,sigma,error_code1);
   cdf2=MathCumulativeDistributionNormal(x2,mu,sigma,error_code2);
//--- comprobamos el código de los errores
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- calculamos la probabilidad de una magnitud aleatoria en el rango
      probability=cdf2-cdf1;
      //--- mostramos el resultado
      PrintFormat("1. Calcular en el rango %.5f<x<%.5f la probabilidad de una magnitud
      PrintFormat("  Respuesta: Probability = %5.8f",probability);
     }

//--- Encontramos el intervalo de valores de la magnitud aleatoria x, que corresponde
   probability=0.95; //  establecemos la probabilidad confiada
//--- establecemos la probabilidad en los límites del intervalo
   double p1=(1.0-probability)*0.5;
   double p2=probability+(1.0-probability)*0.5;
//--- calculamos los límites del intervalo
   x1=MathQuantileNormal(p1,mu,sigma,error_code1);
   x2=MathQuantileNormal(p2,mu,sigma,error_code2);
//--- comprobamos el código de los errores
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- mostramos el resultado
      PrintFormat("2. Para el intervalo confiado = %.2f encontrar el rango de la magni
      PrintFormat("  Respuesta: rango  %5.8f <= x <=%5.8f",x1,x2);
     }

   PrintFormat("3. Calcular los primeros 4 momentos teóricos y calculados de la distri
//--- Generamos una matriz de números aleatorios, calculamos los primeros 4 momentos y
   int data_count=1000000;  // establecemos el número de valores y preparamos la matri
   double data[];
   ArrayResize(data,data_count);
//--- generamos los valores aleatorios y los guardamos en la matriz
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<data_count; i++)
     {
      data[i]=MathRandomNormal(mu,sigma,error_code1);
     }
//--- establecemos el índice del valor incial y el número de datos para el cálculo
   int start=0;
   int count=data_count;
//--- calculamos los primeros 4 momentos de los valores generados
   double mean=MathMean(data,start,count);
   double variance=MathVariance(data,start,count);
   double skewness=MathSkewness(data,start,count);
   double kurtosis=MathKurtosis(data,start,count);
//--- variables para los momentos teóricos
   double normal_mean=0;
   double normal_variance=0;
   double normal_skewness=0;
   double normal_kurtosis=0;
//--- mostramos los valores de los momentos calculados
   PrintFormat("            Mean           Variance          Skewness           Kurtos
   PrintFormat("Calculated  %.10f   %.10f      %.10f      %.10f",mean,variance,skewnes
//--- calculamos los valores teóricos de los momentos y comparamos con los obtenidos
   if(MathMomentsNormal(mu,sigma,normal_mean,normal_variance,normal_skewness,normal_ku
     {
      PrintFormat("Theoretical %.10f   %.10f      %.10f       %.10f",normal_mean,norma
      PrintFormat("Difference  %.10f   %.10f      %.10f       %.10f",mean-normal_mean,
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Características estadísticas

Este grupo de funciones calcula las características estadísticas de los elemenos de la matriz:

- media,
- varianza,
- coeficiente de asimetría,
- curtosis,
- mediana,
- desviación media cuadrática
- y estándar

Función	Descripción
MathMean	Calcula el valor medio (primer momento) de los elementos de la matriz
MathVariance	Calcula la varianza (segundo momento) de los elementos de la matriz
MathSkewness	Calcula el coeficiente de asimetría (tercer momento) de los elementos de la matriz
MathKurtosis	Calcula la curtosis (cuarto momento) de los elementos de la matriz
MathMoments	Calcula los primeros 4 momentos (media, varianza, coeficiente de asimetría, curtosis) de los elementos de la matriz
MathMedian	Calcula el valor mediano de los elementos de la matriz
MathStandardDeviation	Calcula la desviación estándar de los elementos de la matriz
MathAverageDeviation	Calcula la desviación media de los elementos de la matriz

© 2000-2025, MetaQuotes Ltd.

MathMean

Calcula el valor medio (primer momento) de los elementos de la matriz. Análogo de mean() en R.

```
double  MathMean(
   const double&  array[]                // matriz con los datos
   );
```

Parámetros

array

[in] Matriz con los datos para el cálculo del valor medio.

start=0

[in] Índice inicial para el cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para el cálculo.

Valor devuelto

Valor medio de los elementos de la matriz. En caso de error, retorna NaN (no numérico).

© 2000-2025, MetaQuotes Ltd.

MathVariance

Calcula la varianza (segundo momento) de los elementos de la matriz. Análogo de var() en R.

```
double  MathVariance(
   const double&  array[]                // matriz con los datos
   );
```

Parámetros

array

[in] Matriz con los datos para el cálculo.

start=0

[in] Índice inicial para el cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para el cálculo.

Valor devuelto

Varianza de los elementos de la matriz. En caso de error, retorna NaN (no numérico).

© 2000-2025, MetaQuotes Ltd.

MathSkewness

Calcula el coeficiente de asimetría (tercer momento) de los elementos de la matriz. Análogo de skewness() en R (biblioteca e1071).

```
double  MathSkewness(
   const double&  array[]                // matriz con los datos
   );
```

Parámetros

array

[in] Matriz con los datos para el cálculo.

start=0

[in] Índice inicial para el cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para el cálculo.

Valor devuelto

Coeficiente de asimetría de los elementos de la matriz. En caso de error, retorna NaN (no numérico).

© 2000-2025, MetaQuotes Ltd.

MathKurtosis

Calcula la curtosis (cuarto momento) de los elementos de la matriz. Análogo de kurtosis() en R (biblioteca e1071).

```
double  MathKurtosis(
   const double&  array[]                // matriz con los datos
   );
```

Parámetros

array

[in] Matriz con los datos para el cálculo.

start=0

[in] Índice inicial para el cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para el cálculo.

Valor devuelto

Curtosis de los elementos de la matriz. En caso de error, retorna NaN (no numérico).

Nota

El cálculo de la curtosis se realiza con respecto a la distribución normal (excess kurtosis=kurtosis-3), es decir, la curtosis excesiva de una distribución normal es igual a cero.

Es positiva si el pico de la distribución cerca del valor esperado es agudo, y negativa, si el pico es plano.

© 2000-2025, MetaQuotes Ltd.

MathMoments

Calcula los primeros 4 momentos (media, varianza, coeficiente de asimetría, curtosis) de los elementos de la matriz.

```
double  MathMoments(
   const double&  array[],              // matriz con los datos
   double&        mean,                 // valor medio (1 momento)
   double&        variance,             // varianza (2 momento)
   double&        skewness,             // coeficiente de asimetría (3 momento)
   double&        kurtosis,             // curtosis (4 momento)
   const int      start=0,              // índice inicial
   const int      count=WHOLE_ARRAY     // número de elementos
   );
```

Parámetros

array

[in] Matriz con los datos para el cálculo.

mean

[out] Variable para el valor medio (1 momento)

variance

[out] Variable para la varianza (2 momento)

skewness

[out] Variable para el coeficiente de asimetría (3 momento)

kurtosis

[out] Variable para la curtosis (4 momento)

start=0

[in] Índice inicial para el cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para el cálculo.

Valor devuelto

Retorna true, si los momentos se han calculado con éxito, si no, false.

Nota

El cálculo de la curtosis se realiza con respecto a la distribución normal (excess kurtosis=kurtosis-3), es decir, la curtosis excesiva de una distribución normal es igual a cero.

Es positiva si el pico de la distribución cerca del valor esperado es agudo, y negativa, si el pico es plano.

© 2000-2025, MetaQuotes Ltd.

MathMedian

Calcula el valor mediano de los elementos de la matriz. Análogo de median() en R.

```
double  MathMedian(
   const double&  array[]                // matriz con los datos
   );
```

Parámetros

array

[in] Matriz con los datos para el cálculo.

start=0

[in] Índice inicial para el cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para el cálculo.

Valor devuelto

Valor mediano de los elementos de la matriz. En caso de error, retorna NaN (no numérico).

© 2000-2025, MetaQuotes Ltd.

MathStandardDeviation

Calcula la desviación estándar de los elementos de la matriz. Análogo de sd() en R.

```
double  MathStandardDeviation(
   const double&  array[]                // matriz con los datos
   );
```

Parámetros

array

[in] Matriz con los datos para el cálculo.

start=0

[in] Índice inicial para el cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para el cálculo.

Valor devuelto

Desviación estándar de los elementos de la matriz. En caso de error, retorna NaN (no numérico).

© 2000-2025, MetaQuotes Ltd.

MathAverageDeviation

Calcula la desviación media de los elementos de la matriz. Análogo de aad() en R.

```
double  MathAverageDeviation(
   const double&  array[]                // matriz con los datos
   );
```

Parámetros

array

[in] Matriz con los datos para el cálculo.

start=0

[in] Índice inicial para el cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para el cálculo.

Valor devuelto

Desviación media de los elementos de la matriz. En caso de error, retorna NaN (no numérico).

© 2000-2025, MetaQuotes Ltd.

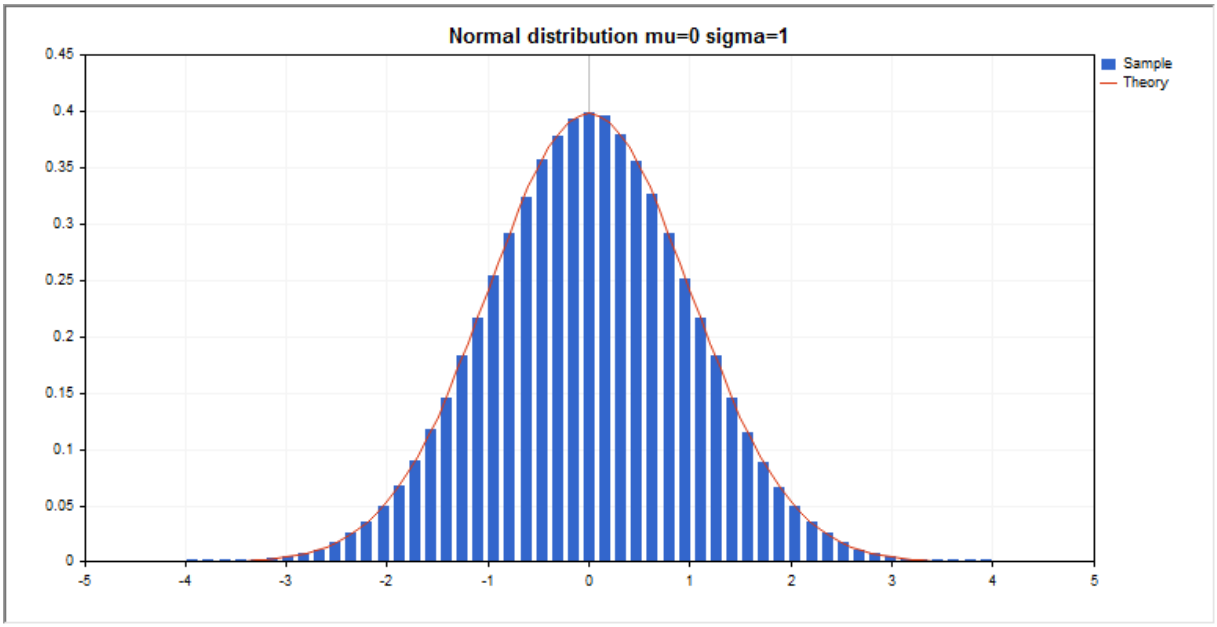
Distribución normal

En este apartado se muestran las funciones para trabajar con la distribución normal. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley normal. La distribución se describe con la siguiente fórmula:

$$f_{Normal}(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

donde:

- x — valor de la magnitud aleatoria
- μ — esperanza matemática
- σ — desviación media cuadrática



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityNormal	Calcula la densidad de probabilidad de la distribución normal
MathCumulativeDistributionNormal	Calcula el valor de la función de la distribución normal de la probabilidad
MathQuantileNormal	Calcula el valor de la función inversa de la distribución normal para la probabilidad indicada
MathRandomNormal	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución normal

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathMomentsNormal	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución normal

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Normal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double mean_value=0;  // esperanza matemática (mean)
input double std_dev=1;     // desviación media cuadrática (standard deviation)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores en la muestra
   int ncells=51;       // número de intervalos en el histograma
   double x[];          // centros de los intervalos del histograma
   double y[];          // número de valores de la muestra que han entrado en el inter
   double data[];       // muestra de valores aleatorios
   double max,min;      // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución normal
   MathRandomNormal(mean_value,std_dev,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNormal(x2,mean_value,std_dev,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Normal distribution mu=%G sigma=%G",mean_value
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
//--- plot all curves
   graphic.CurvePlotAll();
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNormal

Calcula la densidad de la probabilidad de la distribución normal con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNormal(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // parámetro de distribución mean (esperanza matemáti
   const double  sigma,          // parámetro de distribución sigma (desviación media
   const bool    log_mode,       // cálculo del logaritmo del valor
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución normal con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNormal(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // parámetro de distribución mean (esperanza matemáti
   const double  sigma,          // parámetro de distribución sigma (desviación media
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución normal con los parámetros mu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dnorm() en R.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],            // matriz con los valores de la magnitud aleatoria
   const double   mu,             // parámetro de distribución mean (esperanza matemát
   const double   sigma,          // parámetro de distribución sigma (desviación media
   const bool     log_mode,       // cálculo del logaritmo del valor
   double&        result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución normal con los parámetros mu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],            // matriz con los valores de la magnitud aleatoria
   const double   mu,             // parámetro de distribución mean (esperanza matemát
   const double   sigma,          // parámetro de distribución sigma (desviación media
   double&        result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

mu

[in] Parámetro de distribución (esperanza matemática).

sigma

[in] Parámetro de distribución sigma (desviación media cuadrática).

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNormal

Calcula el valor de la función de distribución normal de la probabilidad con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // esperanza matemática
   const double  sigma,          // desviación media cuadrática
   const bool    tail,           // bandera del cálculo de la cola (tail)
   const bool    log_mode,       // cálculo del logaritmo del valor
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función de distribución normal de la probabilidad con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // esperanza matemática
   const double  sigma,          // desviación media cuadrática
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función de distribución normal de la probabilidad con los parámetros mu y sigma para la matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dnorm() en R.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  mu,             // esperanza matemática
   const double  sigma,          // desviación media cuadrática
   const bool    tail,           // bandera del cálculo de la cola (tail)
   const bool    log_mode,       // cálculo del logaritmo del valor
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula el valor de la función de distribución normal de la probabilidad con los parámetros mu y sigma para la matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  mu,             // esperanza matemática
   const double  sigma,          // desviación media cuadrática
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

mu

[in] Parámetro de distribución (esperanza matemática).

sigma

[in] Parámetro de distribución sigma (desviación media cuadrática).

tail

[in] Bandera de cálculo. Si tail=true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNormal

Para la probabilidad dada *probability*, se calcula el valor de la función inversa de distribución normal con los parámetros mu y sigma. En caso de error, retorna NaN.

```
double  MathQuantileNormal(
   const double  probability,    // valor de la probabilidad de una magnitud aleatoria
   const double  mu,             // esperanza matemática
   const double  sigma,          // desviación media cuadrática
   const bool    tail,           // bandera del cálculo de la cola (tail)
   const bool    log_mode,       // cálculo del logaritmo del valor
   int&          error_code      // variable para anotar el código de error
   );
```

Para la probabilidad dada *probability*, se calcula el valor de la función inversa de distribución normal con los parámetros mu y sigma. En caso de error, retorna NaN.

```
double  MathQuantileNormal(
   const double  probability,    // valor de la probabilidad de una magnitud aleatoria
   const double  mu,             // esperanza matemática
   const double  sigma,          // desviación media cuadrática
   int&          error_code      // variable para anotar el código de error
   );
```

Para la matriz de valores de probabilidad *probability*, se calculan los valores de la función inversa de distribución normal con los parámetros mu y sigma. En caso de error, retorna false. Análogo de qnorm() en R.

```
bool  MathQuantileNormal(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  mu,             // esperanza matemática
   const double  sigma,          // desviación media cuadrática
   const bool    tail,           // bandera del cálculo de la cola (tail)
   const bool    log_mode,       // cálculo del logaritmo del valor
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Para la matriz de valores de probabilidad *probability*, se calculan los valores de la función inversa de distribución normal con los parámetros mu y sigma. En caso de error, retorna false.

```
bool  MathQuantileNormal(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  mu,             // esperanza matemática
   const double  sigma,          // desviación media cuadrática
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

mu

[in] Parámetro de distribución (esperanza matemática).

sigma

[in] Parámetro de distribución sigma (desviación media cuadrática).

tail

[in] Bandera de cálculo. Si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para obtener el código de error.

result[]

[out] Variable para obtener los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomNormal

Genera una magnitud pseudoaleatoria, distribuida según la ley normal con los parámetros mu y sigma. En caso de error, retorna NaN.

```
double  MathRandomNormal(
   const double  mu,              // esperanza matemática
   const double  sigma,           // desviación media cuadrática
   int&          error_code       // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley normal con los parámetros mu y sigma. En caso de error, retorna false. Análogo de rnorm() en R.

```
bool  MathRandomNormal(
   const double  mu,              // esperanza matemática
   const double  sigma,           // desviación media cuadrática
   const int     data_count,      // número de datos necesarios
   double&       result[]         // matriz para obtener las magnitudes pseudoaleatoria
   );
```

Parámetros

mu

[in] Parámetro de distribución (esperanza matemática).

sigma

[in] Parámetro de distribución sigma (desviación media cuadrática).

data_count

[in] Número de valores pseudoaleatorios que es necesario obtener.

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNormal

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución normal.

```
double  MathMomentsNormal(
   const double  mu,              // esperanza matemática
   const double  sigma,           // desviación media cuadrática
   double&       mean,            // variable para el valor medio
   double&       variance,        // variable para la varianza
   double&       skewness,        // variable para el coeficiente de asimetría
   double&       kurtosis,        // variable para la curtosis
   int&          error_code       // variable para anotar el código de error
   );
```

Parámetros

mu

[in] Parámetro de distribución (esperanza matemática).

sigma

[in] Parámetro de distribución sigma (desviación media cuadrática).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si los momentos se han calculado con éxito, si no, false.

© 2000-2025, MetaQuotes Ltd.

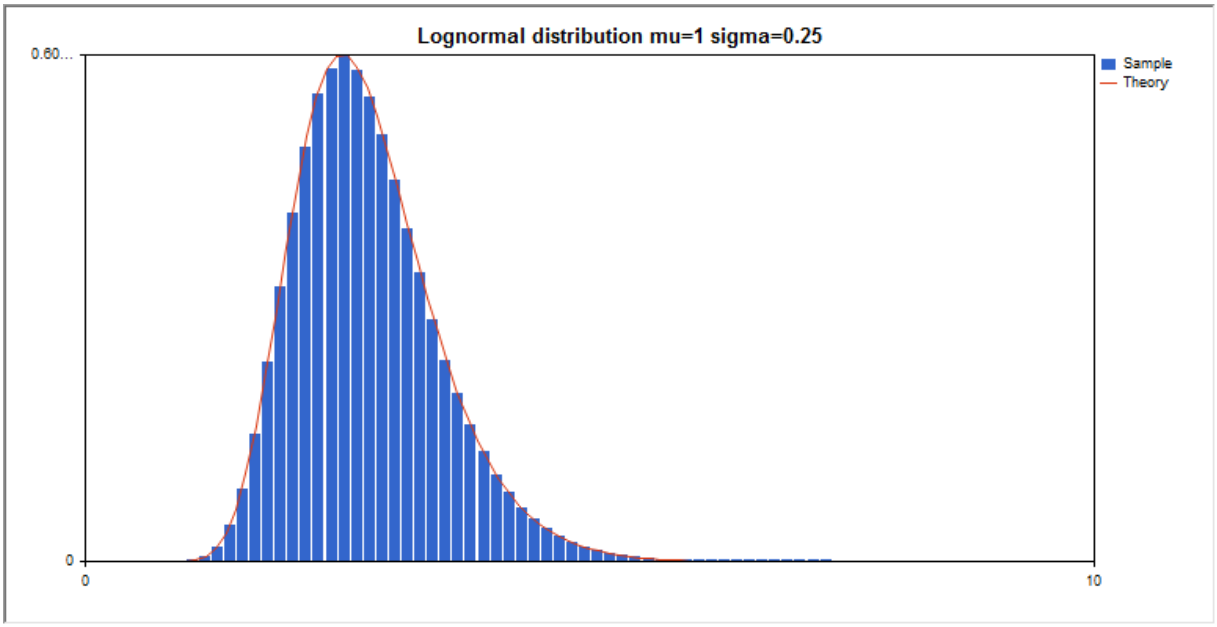
Distribución log-normal

En este apartado se muestran las funciones para trabajar con la distribución log-normal. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley log-normal. La distribución log-normal se describe con la siguiente fórmula:

$$f_{Lognormal}(x|\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$$

donde:

- x — valor de la magnitud aleatoria
- μ — logaritmo de esperanza matemática
- σ — logaritmo de desviación media cuadrática



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityLognormal	Calcula la densidad de probabilidad de la distribución log-normal
MathCumulativeDistributionLognormal	Calcula el valor de la función de la distribución log-normal de la probabilidad
MathQuantileLognormal	Calcula el valor de la función inversa de la distribución log-normal para la probabilidad indicada
MathRandomLognormal	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución log-normal

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathMomentsLognormal	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución log-normal

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Lognormal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double mean_value=1.0;  // logaritmo de esperanza matemática (log mean)
input double std_dev=0.25;    // logaritmo de desviación media cuadrática (log standa
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores en la muestra
   int ncells=51;       // número de intervalos en el histograma
   double x[];          // centros de los intervalos del histograma
   double y[];          // número de valores de la muestra que han entrado en el inte
   double data[];       // muestra de valores aleatorios
   double max,min;      // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución log-normal
   MathRandomLognormal(mean_value,std_dev,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLognormal(x2,mean_value,std_dev,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Lognormal distribution mu=%G sigma=%G",mean_va
   graphic.BackgroundMainSize(16);
//--- desactivamos el escalado automático del eje Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

© 2000-2025, MetaQuotes Ltd.

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityLognormal

Calcula la densidad de la probabilidad de la distribución log-normal con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // logaritmo de esperanza matemática (log mean)
   const double  sigma,          // logaritmo de desviación media cuadrática (log star
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución log-normal con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // logaritmo de esperanza matemática (log mean)
   const double  sigma,          // logaritmo de desviación media cuadrática (log star
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución log-normal con los parámetros mu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna NaN. Análogo de dlnorm() en R.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],            // matriz con los valores de la magnitud aleatoria
   const double   mu,             // logaritmo de esperanza matemática (log mean)
   const double   sigma,          // logaritmo de desviación media cuadrática (log sta
   const bool     log_mode,       // cálculo del logaritmo del valor, si log_mode=true
   double&        result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución log-normal con los parámetros mu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],            // matriz con los valores de la magnitud aleatoria
   const double   mu,             // logaritmo de esperanza matemática (log mean)
   const double   sigma,          // logaritmo de desviación media cuadrática (log sta
   double&        result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

mu

[in] Logaritmo de esperanza matemática (log_mean).

sigma

[in] Logaritmo de desviación media cuadrática (log standard deviation).

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionLognormal

Calcula la distribución log-normal de la probabilidad con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // logaritmo de esperanza matemática (log mean)
   const double  sigma,          // logaritmo de desviación media cuadrática (log star
   const bool    tail,           // bandera para calcular, si true, entonces se calcul
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula la distribución log-normal de la probabilidad con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // logaritmo de esperanza matemática (log mean)
   const double  sigma,          // logaritmo de desviación media cuadrática (log star
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula la distribución log-normal de la probabilidad con los parámetros mu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de plnorm() en R.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // matriz con los valores de la magnitud aleatoria
   const double   mu,            // logaritmo de esperanza matemática (log mean)
   const double   sigma,         // logaritmo de desviación media cuadrática (log sta
   const bool     tail,          // bandera para calcular, si true, entonces se calcu
   const bool     log_mode,      // cálculo del logaritmo del valor, si log_mode=true
   double&        result[]       // matriz para los valores de la función de probabil
   );
```

Calcula la distribución log-normal de la probabilidad con los parámetros mu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // matriz con los valores de la magnitud aleatoria
   const double   mu,            // logaritmo de esperanza matemática (log mean)
   const double   sigma,         // logaritmo de desviación media cuadrática (log sta
   double&        result[]       // matriz para los valores de la función de probabil
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

mu

[in] Logaritmo de esperanza matemática (log_mean).

sigma

[in] Logaritmo de desviación media cuadrática (log standard deviation).

tail

[in] Bandera para calcular, si lower_tail=true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileLognormal

Calcula el valor de la función inversa de distribución log-normal con los parámetros mu y sigma para la probabilidad "probability". En caso de error, retorna NaN.

```
double  MathQuantileLognormal(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  mu,             // logaritmo de esperanza matemática (log mean)
   const double  sigma,          // logaritmo de desviación media cuadrática (log star
   const bool    tail,           // bandera para calcular, si false, entonces el cálcu
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de distribución log-normal con los parámetros mu y sigma para la probabilidad "probability". En caso de error, retorna NaN.

```
double  MathQuantileLognormal(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  mu,             // logaritmo de esperanza matemática (log mean)
   const double  sigma,          // logaritmo de desviación media cuadrática (log star
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de distribución log-normal con los parámetros mu y sigma para una matriz de valores de probabilidad <t1>probability</t1>. En caso de error, retorna false. Análogo de qlnorm() en R.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // matriz con los valores de probabilidad de la magn
   const double   mu,             // logaritmo de esperanza matemática (log mean)
   const double   sigma,          // logaritmo de desviación media cuadrática (log sta
   const bool     tail,           // bandera para calcular, si false, entonces el cálc
   const bool     log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&        result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de distribución log-normal con los parámetros mu y sigma para una matriz de valores de probabilidad probability[]. En caso de error, retorna false.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // matriz con los valores de probabilidad de la magn
   const double   mu,             // logaritmo de esperanza matemática (log mean)
   const double   sigma,          // logaritmo de desviación media cuadrática (log sta
   double&        result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de aparición de una magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

mu

[in] Logaritmo de esperanza matemática (log_mean).

sigma

[in] Logaritmo de desviación media cuadrática (log standard deviation).

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomLognormal

Genera una magnitud pseudoaleatoria, distribuida según la ley log-normal con los parámetros mu y sigma. En caso de error, retorna NaN.

```
double  MathRandomLognormal(
   const double  mu,              // logaritmo de esperanza matemática (log mean)
   const double  sigma,           // logaritmo de desviación media cuadrática (log star
   int&          error_code       // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley log-normal con los parámetros mu y sigma. En caso de error, retorna false. Análogo de rlnorm() en R.

```
double  MathRandomLognormal(
   const double  mu,              // logaritmo de esperanza matemática (log mean)
   const double  sigma,           // logaritmo de desviación media cuadrática (log star
   const int     data_count,      // número de datos necesarios
   double&       result[]         // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

mu

[in] Logaritmo de esperanza matemática (log_mean).

sigma

[in] Logaritmo de desviación media cuadrática (log standard deviation).

data_count

[in] Número de datos necesarios.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz con los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsLognormal

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución log-normal. Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

```
double  MathMomentsLognormal(
   const double  mu,              // logaritmo de esperanza matemática (log mean)
   const double  sigma,           // logaritmo de desviación media cuadrática (log star
   double&       mean,            // variable para el valor medio
   double&       variance,        // variable para la varianza
   double&       skewness,        // variable para el coeficiente de asimetría
   double&       kurtosis,        // variable para la curtosis
   int&          error_code       // variable para anotar el código de error
   );
```

Parámetros

mu

[in] Logaritmo de esperanza matemática (log_mean).

sigma

[in] Logaritmo de desviación media cuadrática (log standard deviation).

mean

[in] Variable para el valor medio.

variance

[out] Variable para la varianza.

skewness

[out] Variable para el coeficiente de asimetría.

kurtosis

[out] Variable para la curtosis.

error code

[out] Variable para anotar el código de error.

Valor devuelto

Retorna true, si los momentos se han calculado con éxito, si no, false.

© 2000-2025, MetaQuotes Ltd.

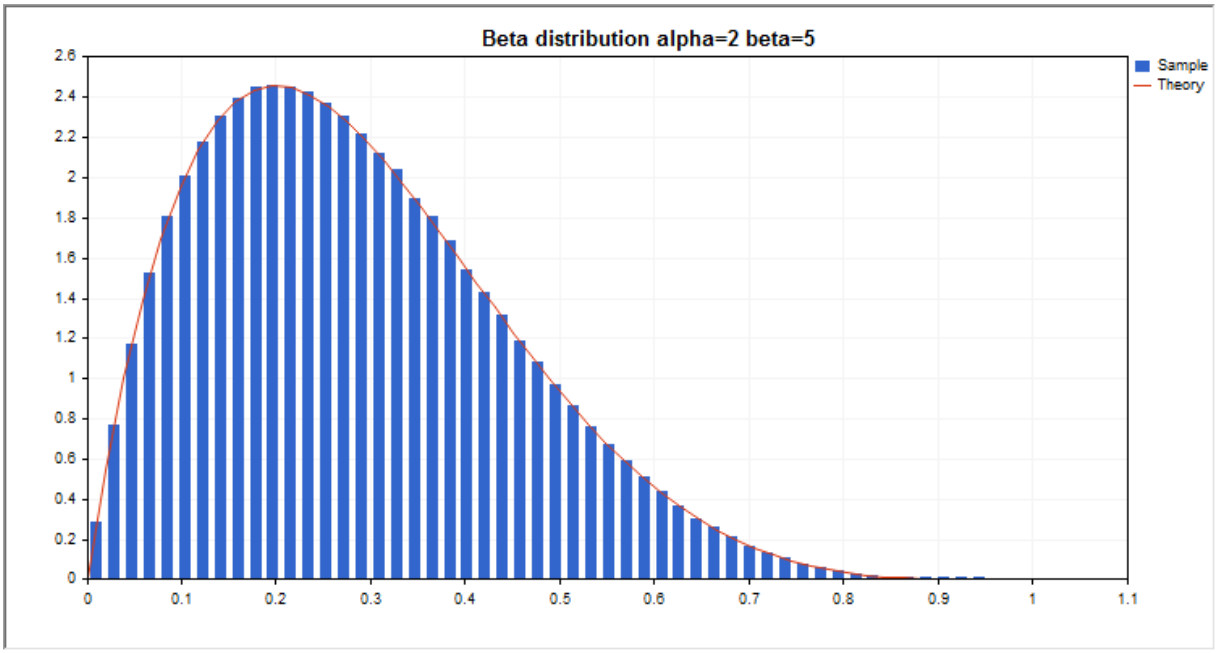
Distribución beta

En este apartado se muestran las funciones para trabajar con la distribución beta. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley consiguiente. La distribución beta se describe con la siguiente fórmula:

$$f_{Beta}(x \mid a,b) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$$

donde:

- x — valor de la magnitud aleatoria
- a — primer parámetro de la distribución beta
- b — segundo parámetro de la distribución beta



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityBeta	Calcula la densidad de probabilidad de la distribución beta
MathCumulativeDistributionBeta	Calcula el valor de la función de la distribución beta de la probabilidad
MathQuantileBeta	Calcula el valor de la función inversa de la distribución beta para la probabilidad indicada
MathRandomBeta	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución beta
MathMomentsBeta	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución beta

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Beta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double alpha=2;   // primer parámetro de la distribución beta (shape1)
input double beta=5;    // segundo parámetro de la distribución beta (shape2)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // número de valores en la muestra
   int ncells=51;          // número de intervalos en el histograma
   double x[];             // centros de los intervalos del histograma
   double y[];             // número de valores de la muestra que han entrado en el inter
   double data[];          // muestra de valores aleatorios
   double max,min;         // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución beta
   MathRandomBeta(alpha,beta,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityBeta(x2,alpha,beta,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Beta distribution alpha=%G beta=%G",alpha,beta
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityBeta

Calcula la densidad de la probabilidad de la distribución beta con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityBeta(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución beta con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityBeta(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución beta con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dbeta() en R.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si
   double&       result[]        // matriz para el valor de la función de densidad de
   );
```

Calcula la densidad de la probabilidad de la distribución beta con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   double&       result[]        // matriz para el valor de la función de densidad de
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

a

[in] Primer parámetro de la distribución beta (shape 1).

b

[in] Segundo parámetro de la distribución beta (shape 2)

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para el valor de la función de densidad de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionBeta

Calcula la distribución de la probabilidad de la distribución beta con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const bool    tail,           // Bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de la distribución beta con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de la distribución beta con los parámetros a y b para una matriz de magnitudes aleatorias x[] En caso de error, retorna false. Análogo de pbeta() en R.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const bool    tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para el valor de la función de probabilidad
   );
```

Calcula la distribución de la probabilidad de la distribución beta con los parámetros a y b para una matriz de magnitudes aleatorias x[] En caso de error, retorna false.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

a

[in] Primer parámetro de la distribución beta (shape 1).

b

[in] Segundo parámetro de la distribución beta (shape 2)

tail

[in] Bandera para calcular, si lower_tail=true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileBeta

Para la probabilidad *probability* el valor de la función inversa de la distribución beta con los parámetros a y b. En caso de error, retorna NaN.

```
double  MathQuantileBeta(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Para la probabilidad *probability* el valor de la función inversa de la distribución beta con los parámetros a y b. En caso de error, retorna NaN.

```
double  MathQuantileBeta(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula para la matriz de valores de la probabilidad *probability[]* el valor de la función inversa de la distribución beta con los parámetros a y b. En caso de error, retorna false. Análogo de qbeta() en R.

```
double  MathQuantileBeta(
   const double&  probability[], // matriz con los valores de probabilidad de la magni
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula para la matriz de valores de la probabilidad *probability[]* el valor de la función inversa de la distribución beta con los parámetros a y b. En caso de error, retorna false.

```
bool  MathQuantileBeta(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

a

[in] Primer parámetro de la distribución beta (shape1).

b

[in] Segundo parámetro de la distribución beta (shape2).

tail

[in] Bandera para calcular, si lower_tail=false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomBeta

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución beta con los parámetros a y b. En caso de error, retorna NaN.

```
double  MathRandomBeta(
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   int&          error_code      // variable para el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución beta con los parámetros a y b. En caso de error, retorna false. Análogo de rbeta() en R.

```
bool  MathRandomBeta(
   const double  a,               // primer parámetro de la distribución beta (shape1)
   const double  b,               // segundo parámetro de la distribución beta (shape2)
   const int     data_count,      // número de datos necesarios
   double&       result[]         // matriz para obtener las magnitudes pseudoaleatoria
   );
```

Parámetros

a

[in] Primer parámetro de la distribución beta (shape1)

b

[in] Segundo parámetro de la distribución beta (shape2).

data_count

[in] Número de valores pseudoaleatorios que es necesario obtener.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsBeta

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución beta.

```
double  MathMomentsBeta(
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

a

[in] Primer parámetro de la distribución beta (shape1).

b

[in] Segundo parámetro de la distribución beta (shape2).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si los momentos se han calculado con éxito, si no, false.

© 2000-2025, MetaQuotes Ltd.

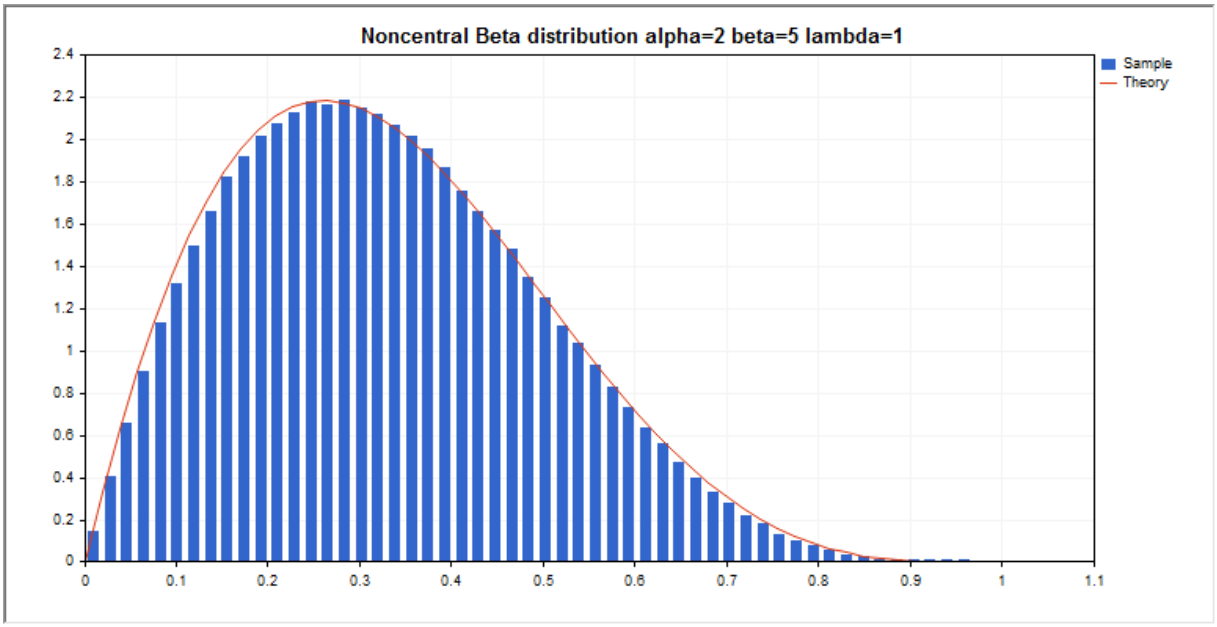
Distribución beta no central

En este apartado se muestran las funciones para trabajar con la distribución beta no central. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley consiguiente. La distribución beta no central se describe con la siguiente fórmula:

$$f_{NoncentralBeta}(x|a,b,\lambda)=\sum_{r=0}^{\infty}e^{-\frac{\lambda}{2}}\frac{\left(\frac{\lambda}{2}\right)^r}{r!}\frac{x^{a+r-1}(1-x)^{b-1}}{B(a+r,b)}$$

donde:

- x — valor de la magnitud aleatoria
- a — primer parámetro de la distribución beta
- b — segundo parámetro de la distribución beta
- λ — parámetro de no-centralidad



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityNoncentralBeta	Calcula la densidad de probabilidad de la distribución beta no central
MathCumulativeDistributionNoncentralBeta	Calcula el valor de la función de la distribución beta no central de la probabilidad
MathQuantileNoncentralBeta	Calcula el valor de la función inversa de la distribución beta no central para la probabilidad indicada

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathRandomNoncentralBeta	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución beta no central
MathMomentsNoncentralBeta	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución beta no central

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralBeta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double a_par=2;     // primer parámetro de la distribución beta (shape1)
input double b_par=5;     // segundo parámetro de la distribución beta (shape2)
input double l_par=1;     // parámetro de no-centralidad (lambda)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // número de valores en la muestra
   int ncells=53;        // número de intervalos en el histograma
   double x[];           // centros de los intervalos del histograma
   double y[];           // número de valores de la muestra que han entrado en el inter
   double data[];        // muestra de valores aleatorios
   double max,min;       // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución beta no central
   MathRandomNoncentralBeta(a_par,b_par,l_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralBeta(x2,a_par,b_par,l_par,false,y2);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral Beta distribution alpha=%G beta=%G
                          a_par,b_par,l_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
```

© 2000-2025, MetaQuotes Ltd.

```
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralBeta

Calcula la densidad de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dbeta() en R.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

© 2000-2025, MetaQuotes Ltd.

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

a

[in] Primer parámetro de la distribución beta (shape 1).

b

[in] Segundo parámetro de la distribución beta (shape 2)

lambda

[in] Parámetro de no-centralidad

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralBeta

Calcula la distribución de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si true, entonces se calcula
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de pbeta() en R.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si true, entonces se calcul
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

© 2000-2025, MetaQuotes Ltd.

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

a

[in] Primer parámetro de la distribución beta (shape 1).

b

[in] Segundo parámetro de la distribución beta (shape 2)

lambda

[in] Parámetro de no-centralidad

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNoncentralBeta

Calcula el valor de la función inversa de distribución de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para la probabilidad de aparición de un valor de la magnitud aleatoria "probability". En caso de error, retorna NaN.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de distribución de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para la probabilidad de aparición de un valor de la magnitud aleatoria "probability". En caso de error, retorna NaN.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de distribución de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para una matriz de valores de probabilidad probability[]. En caso de error, retorna false. Análogo de qbeta() en R.

```
double  MathQuantileNoncentralBeta(
   const double&  probability[],  // matriz con los valores de probabilidad de la magn
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de distribución de la probabilidad de la distribución beta no central con los parámetros a, b y lambda para una matriz de valores de probabilidad probability[]. En caso de error, retorna false.

```
bool  MathQuantileNoncentralBeta(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
```

© 2000-2025, MetaQuotes Ltd.

```
   const double  lambda,          // parámetro de no-centralidad
   double&       result[]         // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

a

[in] Primer parámetro de la distribución beta (shape1).

b

[in] Segundo parámetro de la distribución beta (shape2).

lambda

[in] Parámetro de no-centralidad.

tail

[in] bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralBeta

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución beta no central con los parámetros a, b y lambda. En caso de error, retorna NaN.

```
double  MathRandomNoncentralBeta(
   const double  a,             // primer parámetro de la distribución beta (shape1)
   const double  b,             // segundo parámetro de la distribución beta (shape2)
   const double  lambda,        // parámetro de no-centralidad
   int&          error_code     // variable para el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución beta no central con los parámetros a, b y lambda. En caso de error, retorna false. Análogo de rbeta() en R.

```
bool  MathRandomNoncentralBeta(
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz para obtener las magnitudes pseudoaleatoria
   );
```

Parámetros

a

[in] Primer parámetro de la distribución beta (shape1)

b

[in] Segundo parámetro de la distribución beta (shape2).

lambda

[in] Parámetro de no-centralidad

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralBeta

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución beta no central con los parámetros a, b y lambda.

```
double  MathMomentsNoncentralBeta(
   const double  a,              // primer parámetro de la distribución beta (shape1)
   const double  b,              // segundo parámetro de la distribución beta (shape2)
   const double  lambda,         // parámetro de no-centralidad
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

a

[in] Primer parámetro de la distribución beta (shape1).

b

[in] Segundo parámetro de la distribución beta (shape2).

lambda

[in] Parámetro de no-centralidad

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

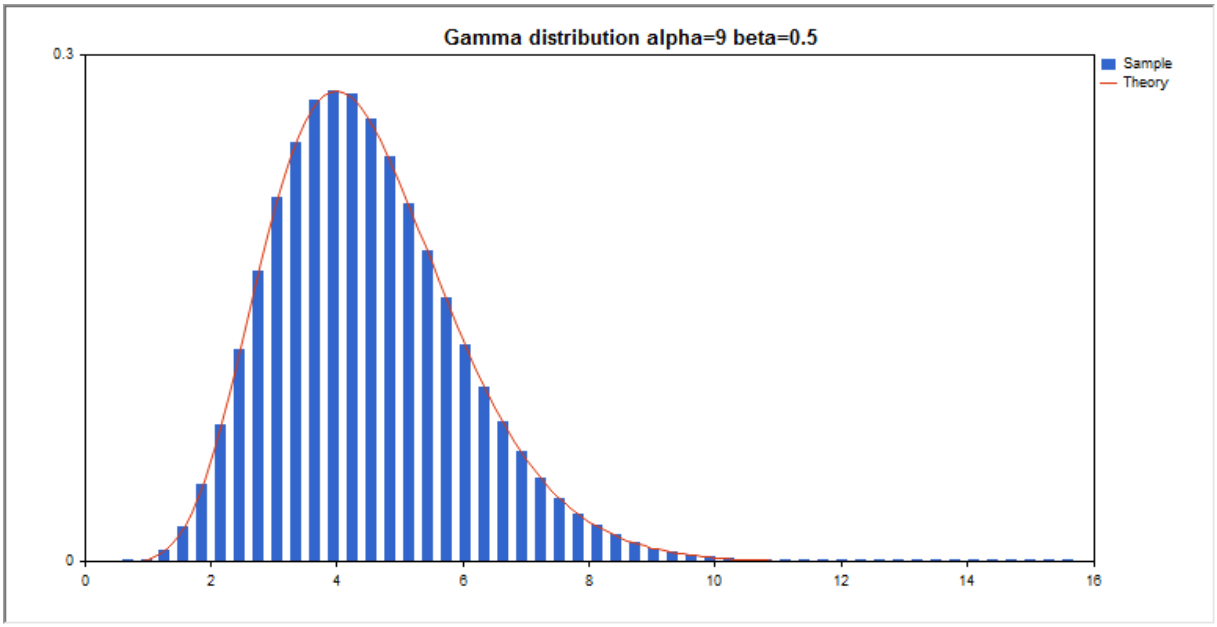
Distribución gamma

En este apartado se muestran las funciones para trabajar con la distribución gamma. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley consiguiente. La distribución gamma se describe con la siguiente fórmula:

$$f_{Gamma}(x|a,b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-\frac{x}{b}}$$

donde:

- x — valor de la magnitud aleatoria
- a — primer parámetro de la distribución
- b — segundo parámetro de la distribución



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityGamma	Calcula la densidad de probabilidad de la distribución gamma
MathCumulativeDistributionGamma	Calcula el valor de la función de la distribución gamma de la probabilidad
MathQuantileGamma	Calcula el valor de la función inversa de la distribución gamma para la probabilidad indicada
MathRandomGamma	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución beta
MathMomentsGamma	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución gamma

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Gamma.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double alpha=9;   // primer parámetro de la distribución gamma (shape)
input double beta=0.5;  // segundo parámetro de la distribución gamma (scale)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // número de valores en la muestra
   int ncells=51;        // número de intervalos en el histograma
   double x[];           // centros de los intervalos del histograma
   double y[];           // número de valores de la muestra que han entrado en el inter
   double data[];        // muestra de valores aleatorios
   double max,min;       // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución gamma
   MathRandomGamma(alpha,beta,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityGamma(x2,alpha,beta,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Gamma distribution alpha=%G beta=%G",alpha,bet
   graphic.BackgroundMainSize(16);
//--- desactivamos el escalado automático del eje Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,1));
   graphic.YAxis().Min(0);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityGamma

Calcula la densidad de la probabilidad de la distribución gamma con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityGamma(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución gamma con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityGamma(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución gamma con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dgamma() en R.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución gamma con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

a

[in] Primer parámetro de la distribución (shape).

b

[in] Segundo parámetro de la distribución (scale).

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionGamma

Calcula la distribución gamma de la probabilidad con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución gamma de la probabilidad con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución gamma de la probabilidad con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de pgamma() en R.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución gamma de la probabilidad con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

a

[in] Primer parámetro de la distribución (shape).

b

[in] Segundo parámetro de la distribución (scale)

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileGamma

Calcula el valor de la función inversa de la distribución gamma con los parámetros a y b para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileGamma(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución gamma con los parámetros a y b para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileGamma(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución gamma con los parámetros a y b para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qgamma() en R.

```
double  MathQuantileGamma(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución gamma con los parámetros a y b para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileGamma(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

a

[in] Primer parámetro de la distribución (shape).

b

[in] Segundo parámetro de la distribución (scale).

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomGamma

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución gamma con los parámetros a y b. En caso de error, retorna NaN.

```
double  MathRandomGamma(
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   int&          error_code      // variable para el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución gamma con los parámetros a y b. En caso de error, retorna false. Análogo de rgamma() en R.

```
bool  MathRandomGamma(
   const double  a,               // primer parámetro de la distribución (shape)
   const double  b,               // segundo parámetro de la distribución (scale)
   const int     data_count,      // número de datos necesarios
   double&       result[]         // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

a

[in] Primer parámetro de la distribución (shape).

b

[in] Segundo parámetro de la distribución (scale).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsGamma

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución gamma con los parámetros a, b.

```
double  MathMomentsGamma(
   const double  a,              // primer parámetro de la distribución (shape)
   const double  b,              // segundo parámetro de la distribución (scale)
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

a

[in] Primer parámetro de la distribución (shape).

b

[in] Segundo parámetro de la distribución (scale).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

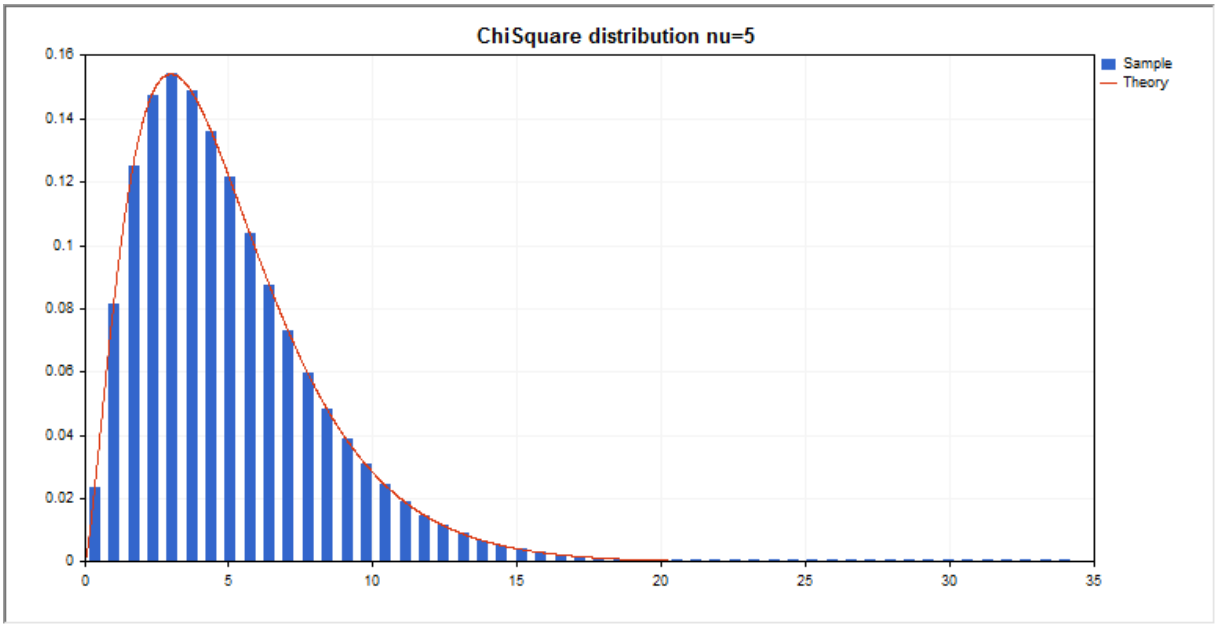
Distribución chi-cuadrado

En este apartado se muestran las funciones para trabajar con la distribución chi-cuadrado. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley consiguiente. La distribución chi-cuadrado se describe con la siguiente fórmula:

$$f_{Chi-Square}(x|\nu) = \frac{x^{\frac{(\nu-2)}{2}} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)}$$

donde:

- x — valor de la magnitud aleatoria
- ν — número de grados de libertad



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityChiSquare	Calcula la densidad de probabilidad de la distribución chi-cuadrado
MathCumulativeDistributionChiSquare	Calcula el valor de la función de la distribución de la probabilidad chi-cuadrado
MathQuantileChiSquare	Calcula el valor de la función inversa de la distribución chi-cuadrado para la probabilidad indicada
MathRandomChiSquare	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución chi-cuadrado

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathMomentsChiSquare	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución chi-cuadrado

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\ChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_par=5;     // número de grados de libertad
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // número de valores en la muestra
   int ncells=51;          // número de intervalos en el histograma
   double x[];             // centros de los intervalos del histograma
   double y[];             // número de valores de la muestra que han entrado en el inter
   double data[];          // muestra de valores aleatorios
   double max,min;         // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución chi-cuadrado
   MathRandomChiSquare(nu_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityChiSquare(x2,nu_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("ChiSquare distribution nu=%G ",nu_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityChiSquare

Calcula la densidad de la distribución de la probabilidad de chi-cuadrado con el parámetro nu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la distribución de la probabilidad de chi-cuadrado con el parámetro nu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la distribución de la probabilidad de chi-cuadrado con el parámetro nu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dchisq() en R.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la distribución de la probabilidad de chi-cuadrado con el parámetro nu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad)

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionChiSquare

Calcula la distribución de la probabilidad de chi-cuadrado con el parámetro nu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de chi-cuadrado con el parámetro nu para una magnitud aleatoria x. En caso de error, retorna NaN

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de chi-cuadrado con el parámetro nu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de pchisq() en R.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución de la probabilidad de chi-cuadrado con el parámetro nu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileChiSquare

Calcula el valor de la función inversa de la distribución de probabilidad chi-cuadrado para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileChiSquare(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución de probabilidad chi-cuadrado para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileChiSquare(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución de probabilidad chi-cuadrado para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qchisq() en R.

```
double  MathQuantileChiSquare(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución de probabilidad chi-cuadrado para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileChiSquare(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomChiSquare

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución chi-cuadrado con el parámetro nu. En caso de error, retorna NaN.

```
double  MathRandomChiSquare(
   const double  nu,            // parámetro de distribución (número de grados de libe
   int&          error_code     // variable para el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución chi-cuadrado con el parámetro nu. En caso de error, retorna false. Análogo de rchisq() en R.

```
bool  MathRandomChiSquare(
   const double  nu,             // parámetro de distribución (número de grados de lib
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

nu

[in] Parámetro de distribución (número de grados de libertad).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsChiSquare

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución chi-cuadrado con el parámetro nu.

```
double  MathMomentsChiSquare(
   const double  nu,              // parámetro de distribución (número de grados de lib
   double&       mean,            // variable para el valor medio
   double&       variance,        // variable para la varianza
   double&       skewness,        // variable para el coeficiente de asimetría
   double&       kurtosis,        // variable para la curtosis
   int&          error_code       // variable para el código de error
   );
```

Parámetros

nu

[in] Parámetro de distribución (número de grados de libertad).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

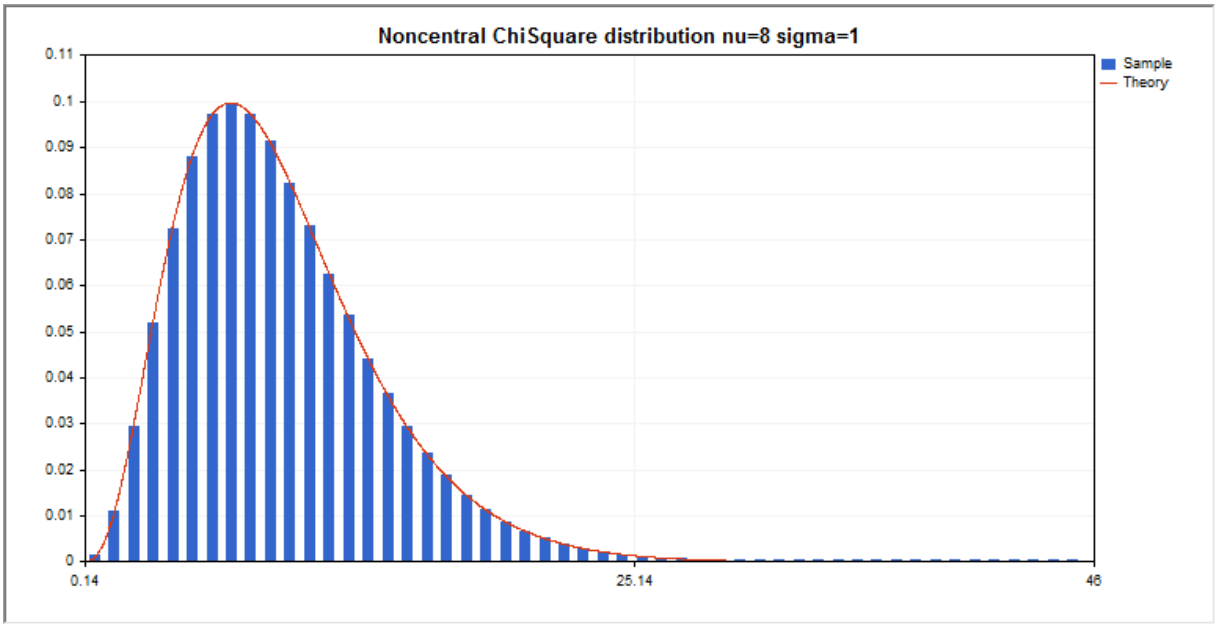
Distribución chi-cuadrado no central

En este apartado se muestran las funciones para trabajar con la distribución chi-cuadrado no central. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley consiguiente. La distribución chi-cuadrado no central se describe con la siguiente fórmula:

$$f_{NoncentralChiSquare}(x|\nu,\sigma) = \frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{1}{2}\right)} x^{\frac{\nu}{2}-1} e^{-\frac{(x+\sigma)}{2}} \sum_{r=0}^{\infty} \frac{(\lambda x)^r}{(2r)!} \frac{\Gamma\left(\frac{1}{2}+r\right)}{\Gamma\left(\frac{\nu}{2}+r\right)}$$

donde:

- x — valor de la magnitud aleatoria
- ν — número de grados de libertad
- σ — parámetro de no-centralidad



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityNoncentralChiSquare	Calcula la densidad de probabilidad de la distribución chi-cuadrado no central
MathCumulativeDistributionNoncentralChiSquare	Calcula el valor de la función de la distribución de la probabilidad chi-cuadrado no central
MathQuantileNoncentralChiSquare	Calcula el valor de la función inversa de la distribución chi-cuadrado no central para la probabilidad indicada

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathRandomNoncentralChiSquare	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución chi-cuadrado no central
MathMomentsNoncentralChiSquare	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución chi-cuadrado no central

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_par=8;     // número de grados de libertad
input double si_par=1;     // parámetro de no-centralidad
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // número de valores en la muestra
   int ncells=51;          // número de intervalos en el histograma
   double x[];             // centros de los intervalos del histograma
   double y[];             // número de valores de la muestra que han entrado en el inter
   double data[];          // muestra de valores aleatorios
   double max,min;         // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución chi-cuadrado no central
   MathRandomNoncentralChiSquare(nu_par,si_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralChiSquare(x2,nu_par,si_par,false,y2);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral ChiSquare distribution nu=%G sigma=
   graphic.BackgroundMainSize(16);
//--- desactivamos el escalado automático del eje X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(NormalizeDouble(max,0));
   graphic.XAxis().Min(min);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralChiSquare

Calcula la densidad de la probabilidad de la distribución chi-cuadrado no central con los parámetros nu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const double  sigma,          // parámetro de no-centralidad
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución chi-cuadrado no central con los parámetros nu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const double  sigma,          // parámetro de no-centralidad
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución chi-cuadrado no central con los parámetros nu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dchisq() en R.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  sigma,          // parámetro de no-centralidad
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución chi-cuadrado no central con el parámetro nu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  sigma,          // parámetro de no-centralidad
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

sigma

[in] Parámetro de no-centralidad.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralChiSquare

Calcula la distribución de la probabilidad de la distribución chi-cuadrado no central con los parámetros nu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const double  sigma,          // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de la distribución chi-cuadrado no central con los parámetros nu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const double  sigma,          // parámetro de no-centralidad
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de la distribución chi-cuadrado no central con los parámetros nu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de pchisq() en R.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  sigma,          // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución de la probabilidad de la distribución chi-cuadrado no central con los parámetros nu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  sigma,          // parámetro de no-centralidad
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

sigma

[in] Parámetro de no-centralidad.

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNoncentralChiSquare

Calcula el valor de la función inversa de distribución chi-cuadrado no central con los parámetros nu y sigma para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  sigma,          // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de distribución chi-cuadrado no central con los parámetros nu y sigma para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  sigma,          // parámetro de no-centralidad
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de distribución chi-cuadrado no central con los parámetros nu y sigma para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qchisq() en R.

```
double  MathQuantileNoncentralChiSquare(
   const double&  probability[],  // matriz con los valores de probabilidad de la magn
   const double  nu,              // parámetro de distribución (número de grados de li
   const double  sigma,           // parámetro de no-centralidad
   const bool    tail,            // bandera para calcular, si lower_tail=false, entor
   const bool    log_mode,        // bandera para calcular, si log_mode=true, entonces
   double&       result[]         // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución de probabilidad chi-cuadrado no central para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileNoncentralChiSquare(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  sigma,          // parámetro de no-centralidad
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

sigma

[in] Parámetro de no-centralidad.

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralChiSquare

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución chi-cuadrado no central con los parámetros nu y sigma. En caso de error, retorna NaN.

```
double  MathRandomNoncentralChiSquare(
   const double  nu,            // parámetro de distribución (número de grados de libe
   const double  sigma,         // parámetro de no-centralidad
   int&          error_code     // variable para el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución chi-cuadrado no central con los parámetros nu y sigma. En caso de error, retorna false. Análogo de rchisq() en R.

```
bool  MathRandomNoncentralChiSquare(
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  sigma,          // parámetro de no-centralidad
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

nu

[in] Parámetro de distribución (número de grados de libertad).

sigma

[in] Parámetro de no-centralidad.

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralChiSquare

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución chi-cuadrado no central con los parámetros nu y sigma.

```
double  MathMomentsNoncentralChiSquare(
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  sigma,          // parámetro de no-centralidad
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

nu

[in] Parámetro de distribución (número de grados de libertad).

sigma

[in] Parámetro de no-centralidad.

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

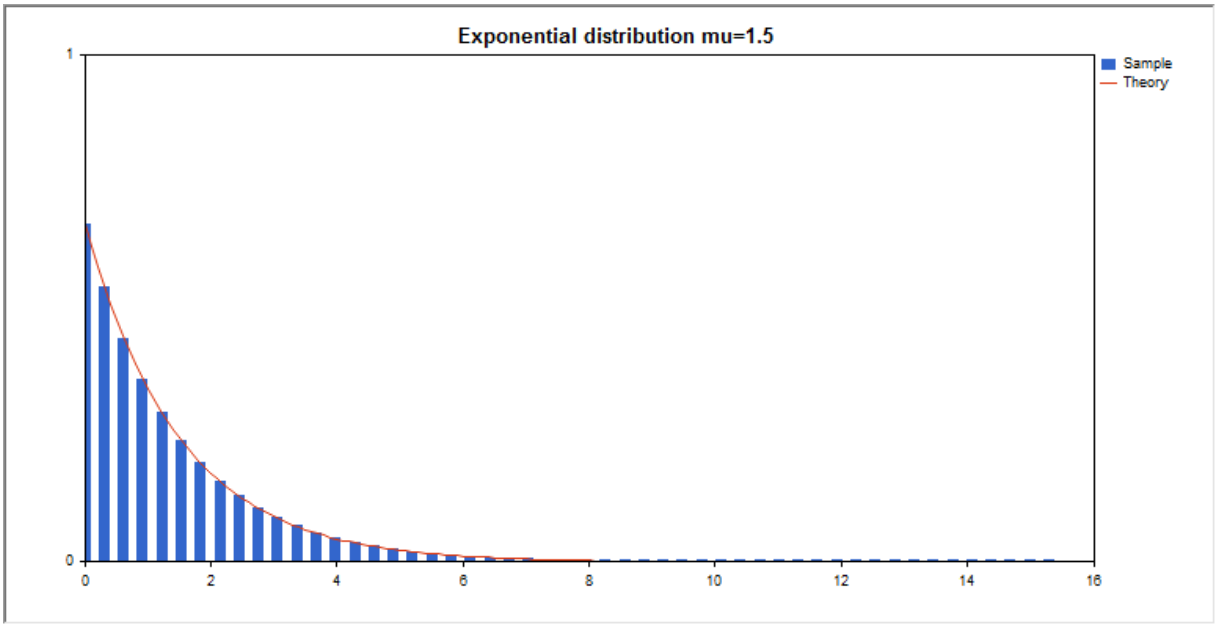
Exponencial

En este apartado se muestran las funciones para trabajar con la distribución exponencial. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley exponencial. La distribución exponencial se describe con la siguiente fórmula:

$$f_{Exponential}(x \mid \mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

donde:

- x — valor de la magnitud aleatoria
- μ — esperanza matemática



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityExponential	Calcula la densidad de probabilidad de la distribución exponencial
MathCumulativeDistributionExponential	Calcula el valor de la función de la distribución exponencial de la probabilidad
MathQuantileExponential	Calcula el valor de la función inversa de la distribución exponencial para la probabilidad indicada
MathRandomExponential	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución exponencial
MathMomentsExponential	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución exponencial.

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Exponential.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double mu_par=1.5;     // número de grados de libertad
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // número de valores en la muestra
   int ncells=51;        // número de intervalos en el histograma
   double x[];           // centros de los intervalos del histograma
   double y[];           // número de valores de la muestra que han entrado en el inter
   double data[];        // muestra de valores aleatorios
   double max,min;       // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución exponencial
   MathRandomExponential(mu_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityExponential(x2,mu_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Exponential distribution mu=%G ",mu_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
```

© 2000-2025, MetaQuotes Ltd.

```
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityExponential

Calcula la densidad de la probabilidad de la distribución exponencial con el parámetro mu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityExponential(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // parámetro de distribución (esperanza matemática)
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución exponencial con el parámetro mu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityExponential(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // parámetro de distribución (esperanza matemática)
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución exponencial con el parámetro mu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dexp() en R.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  mu,             // parámetro de distribución (esperanza matemática)
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución exponencial con el parámetro mu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  mu,             // parámetro de distribución (esperanza matemática)
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

mu

[in] Parámetro de distribución (esperanza matemática)

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionExponential

Calcula la función exponencial de la probabilidad con el parámetro mu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // parámetro de distribución (esperanza matemática
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la función exponencial de la probabilidad con el parámetro mu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // parámetro de distribución (esperanza matemática
   int&          error_code      // variable para el código de error
   );
```

Calcula la función exponencial de la probabilidad con el parámetro mu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de pexp() en R.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  mu,             // parámetro de distribución (esperanza matemática)
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la función exponencial de la probabilidad con el parámetro mu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  mu,             // parámetro de distribución (esperanza matemática)
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

mu

[in] Parámetro de distribución (esperanza matemática).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileExponential

Calcula el valor de la función inversa de la distribución exponencial de probabilidad con el parámetro mu para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileExponential(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  mu,             // parámetro de distribución (esperanza matemática)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución exponencial de probabilidad con el parámetro mu para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileExponential(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  mu,             // parámetro de distribución (esperanza matemática)
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución exponencial de probabilidad con el parámetro mu para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qexp() en R.

```
double  MathQuantileExponential(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  mu,             // parámetro de distribución (esperanza matemática)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución exponencial de probabilidad con el parámetro mu para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileExponential(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  mu,             // parámetro de distribución (esperanza matemática)
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

mu

© 2000-2025, MetaQuotes Ltd.

[in] Parámetro de distribución (esperanza matemática).

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomExponential

Genera una magnitud pseudoaleatoria, distribuida según la ley de distribución exponencial con el parámetro mu. En caso de error, retorna NaN.

```
double  MathRandomExponential(
   const double  mu,            // parámetro de distribución (esperanza matemática
   int&          error_code     // variable para el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de distribución exponencial con el parámetro mu. En caso de error, retorna false. Análogo de rexp() en R.

```
bool  MathRandomExponential(
   const double  mu,             // parámetro de distribución (esperanza matemática)
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

mu

[in] Parámetro de distribución (esperanza matemática).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsExponential

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución exponencial con el parámetro mu.

```
double  MathMomentsExponential(
   const double  mu,              // parámetro de distribución (esperanza matemática)
   double&       mean,            // variable para el valor medio
   double&       variance,        // variable para la varianza
   double&       skewness,        // variable para el coeficiente de asimetría
   double&       kurtosis,        // variable para la curtosis
   int&          error_code       // variable para el código de error
   );
```

Parámetros

mu

[in] Parámetro de distribución (esperanza matemática).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

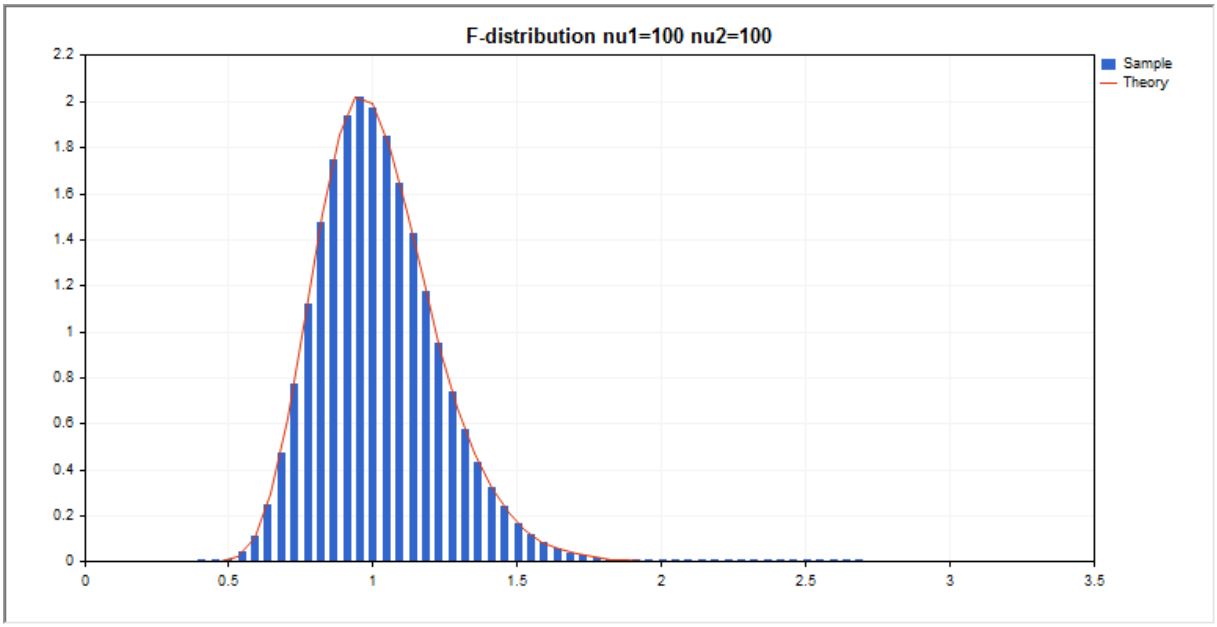
Distribución F

En este apartado se muestran las funciones para trabajar con la distribución F. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley de Fisher. La distribución F se describe con la siguiente fórmula:

$$f_F(x \mid \nu_1, \nu_2) = \frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}\right)}{\Gamma\left(\frac{\nu_1}{2}\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_1}{2}} \frac{x^{\frac{\nu_1-2}{2}}}{\left(1+\left(\frac{\nu_1}{\nu_2}\right)x\right)^{\frac{\nu_1+\nu_2}{2}}}$$

donde:

- x — valor de la magnitud aleatoria
- ν1 — primer parámetro de distribución (número de grados de libertad)
- ν2 — segundo parámetro de distribución (número de grados de libertad)



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityF	Calcula la densidad de probabilidad de la distribución F
MathCumulativeDistributionF	Calcula el valor de la función de la distribución F de la probabilidad
MathQuantileF	Calcula el valor de la función inversa de la distribución F para la probabilidad indicada
MathRandomF	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley F

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathMomentsF	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución F

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\F.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_1=100;     // primer número de grados de libertad
input double nu_2=100;     // segundo número de grados de libertad
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // número de valores en la muestra
   int ncells=51;        // número de intervalos en el histograma
   double x[];           // centros de los intervalos del histograma
   double y[];           // número de valores de la muestra que han entrado en el inter
   double data[];        // muestra de valores aleatorios
   double max,min;       // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución F
   MathRandomF(nu_1,nu_2,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityF(x2,nu_1,nu_2,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("F-distribution nu1=%G nu2=%G",nu_1,nu_2));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(4);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityF

Calcula la densidad de la probabilidad de la distribución F de Fisher con los parámetros nu1 y nu2 para la magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityF(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grados
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución F de Fisher con los parámetros nu1 y nu2 para la magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityF(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grados
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución F de Fisher con los parámetros nu1 y nu2 para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de df() en R.

```
bool  MathProbabilityDensityF(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución F de Fisher con los parámetros nu1 y nu2 para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityF(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

nu1

[in] Primer parámetro de distribución (número de grados de libertad).

nu2

[in] Segundo parámetro de distribución (número de grados de libertad).

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionF

Calcula la distribución de la probabilidad según la ley de la distribución F de Fisher con los parámetros nu1 y nu2 para la magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionF(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grados
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad según la ley de la distribución F de Fisher con los parámetros nu1 y nu2 para la magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionF(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grados
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad según la ley de la distribución F de Fisher con los parámetros nu1 y nu2 para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de pf() en R.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución de la probabilidad según la ley de la distribución F de Fisher con los parámetros nu1 y nu2 para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

nu1

[in] Primer parámetro de distribución (número de grados de libertad).

nu2

[in] Segundo parámetro de distribución (número de grados de libertad).

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileF

Calcula el valor de la función inversa de la distribución F de probabilidad con los parámetros nu1 y n2 para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileF(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución F de probabilidad con los parámetros nu1 y n2 para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileF(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución F de probabilidad con los parámetros nu1 y n2 para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qf() en R.

```
double  MathQuantileF(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución F de probabilidad con los parámetros nu1 y n2 para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileF(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

nu1

[in] Primer parámetro de distribución (número de grados de libertad).

nu2

[in] Segundo parámetro de distribución (número de grados de libertad).

tail

[in] Bandera para calcular, si lower_tail=false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomF

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución F de Fisher con los parámetros nu1 y nu2. En caso de error, retorna NaN.

```
double  MathRandomF(
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   int&          error_code     // variable para el código de error
   );
```

Genera magnitudes pseudoaleatorias, según la ley de la distribución F de Fisher con los parámetros nu1 y nu2.. En caso de error, retorna false. Análogo de rf() en R.

```
bool  MathRandomF(
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

nu1

[in] Primer parámetro de distribución (número de grados de libertad).

nu2

[in] Segundo parámetro de distribución (número de grados de libertad).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsF

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución F de Fisher con los parámetros nu1 y nu2.

```
double  MathMomentsF(
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

nu1

[in] Primer parámetro de distribución (número de grados de libertad).

nu2

[in] Segundo parámetro de distribución (número de grados de libertad).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

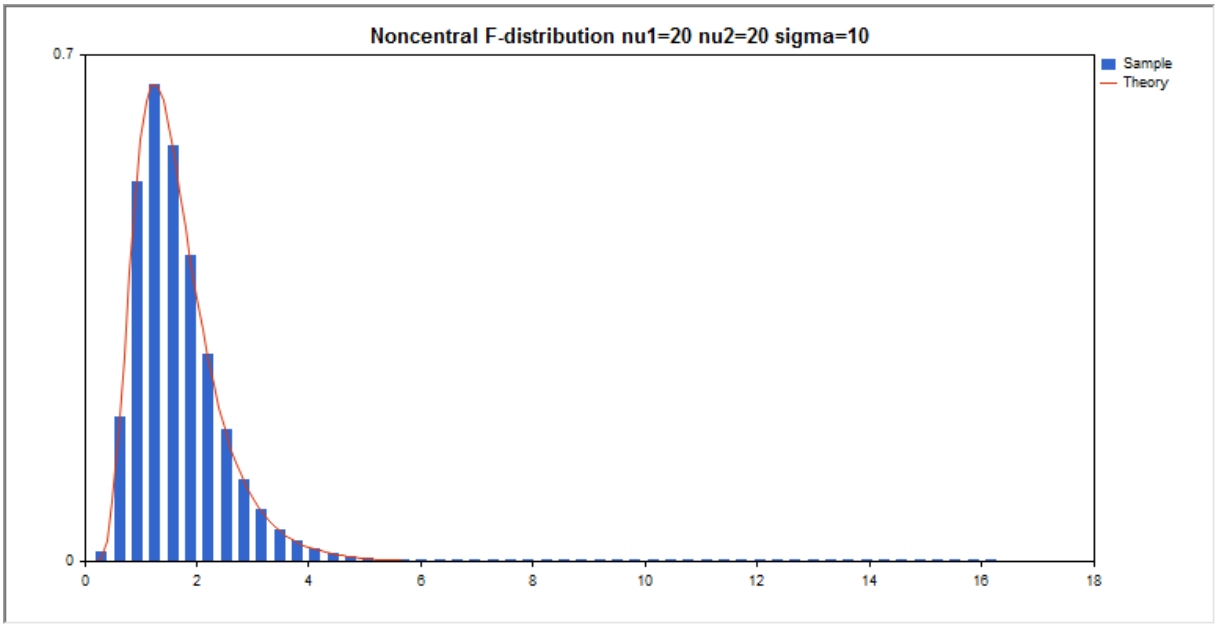
Distribución F no central

En este apartado se muestran las funciones para trabajar con la distribución F no central. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley de distribución de Fisher no central. La distribución F no central se describe con la siguiente fórmula:

$$f_{NoncentralF}(x|\nu_1,\nu_2,\sigma) = e^{-\frac{\sigma}{2}} \sum_{r=0}^{\infty} \frac{1}{r!}\left(\frac{\sigma}{2}\right)^r \frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}+r\right)}{\Gamma\left(\frac{\nu_2}{2}+r\right)\Gamma\left(\frac{\nu_2}{2}\right)} \left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_2}{2}+r} \frac{x^{\frac{\nu_2}{2}-1+r}}{\left(1+\frac{\nu_1}{\nu_2}x\right)^{\frac{\nu_1+\nu_2}{2}+r}}$$

donde:

- x — valor de la magnitud aleatoria
- ν1 — primer parámetro de distribución (número de grados de libertad)
- ν2 — segundo parámetro de distribución (número de grados de libertad)
- σ — parámetro de no-centralidad



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityNoncentralF	Calcula la densidad de probabilidad de la distribución F no central
MathCumulativeDistributionNoncentralF	Calcula el valor de la función de la distribución F no central de la probabilidad
MathQuantileNoncentralF	Calcula el valor de la función inversa de la distribución F no central para la probabilidad indicada

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathRandomNoncentralF	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución Fisher no central
MathMomentsNoncentralF	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución de Fisher no central

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralF.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_1=20;     // primer número de grados de libertad
input double nu_2=20;     // segundo número de grados de libertad
input double sig=10;      // parámetro de no-centralidad
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // número de valores en la muestra
   int ncells=51;        // número de intervalos en el histograma
   double x[];           // centros de los intervalos del histograma
   double y[];           // número de valores de la muestra que han entrado en el inter
   double data[];        // muestra de valores aleatorios
   double max,min;       // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución F no central
   MathRandomNoncentralF(nu_1,nu_2,sig,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
```

© 2000-2025, MetaQuotes Ltd.

```
   MathProbabilityDensityNoncentralF(x2,nu_1,nu_2,sig,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral F-distribution nu1=%G nu2=%G sigma=
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
```

© 2000-2025, MetaQuotes Ltd.

```
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralF

Calcula la densidad de la probabilidad de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para la magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grados
   const double  sigma,          // parámetro de no-centralidad
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para la magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grados
   const double  sigma,          // parámetro de no-centralidad
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de df() en R.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  sigma,          // parámetro de no-centralidad
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  sigma,          // parámetro de no-centralidad
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

© 2000-2025, MetaQuotes Ltd.

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

nu1

[in] Primer parámetro de distribución (número de grados de libertad).

nu2

[in] Segundo parámetro de distribución (número de grados de libertad).

sigma

[in] Parámetro de no-centralidad.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralF

Calcula la distribución de la probabilidad según la ley de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para la magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grados
   const double  sigma,          // parámetro de no-centralidad
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad según la ley de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para la magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grados
   const double  sigma,          // parámetro de no-centralidad
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad según la ley de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de pf() en R.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  sigma,          // parámetro de no-centralidad
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución de la probabilidad según la ley de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  sigma,          // parámetro de no-centralidad
   double&       result[]        // matriz para los valores de la función de probabili
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

nu1

[in] Primer parámetro de distribución (número de grados de libertad).

nu2

[in] Segundo parámetro de distribución (número de grados de libertad).

sigma

[in] Parámetro de no-centralidad.

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileF

Calcula el valor de la función inversa de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileF(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  sigma,          // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileF(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  sigma,          // parámetro de no-centralidad
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qf() en R.

```
double  MathQuantileF(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  sigma,          // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileF(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

© 2000-2025, MetaQuotes Ltd.

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

nu1

[in] Primer parámetro de distribución (número de grados de libertad).

nu2

[in] Segundo parámetro de distribución (número de grados de libertad).

sigma

[in] Parámetro de no-centralidad.

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralF

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma. En caso de error, retorna NaN.

```
double  MathRandomNoncentralF(
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  sigma,          // parámetro de no-centralidad
   int&          error_code      // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias según la ley de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma. En caso de error, retorna false. Análogo de rf() en R.

```
bool  MathRandomNoncentralF(
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  sigma,          // parámetro de no-centralidad
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

nu1

[in] Primer parámetro de distribución (número de grados de libertad).

nu2

[in] Segundo parámetro de distribución (número de grados de libertad).

sigma

[in] Parámetro de no-centralidad.

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralF

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución F de Fisher no central con los parámetros nu1, nu2 y sigma.

```
double  MathMomentsNoncentralF(
   const double  nu1,            // primer parámetro de distribución (número de grados
   const double  nu2,            // segundo parámetro de distribución (número de grad
   const double  sigma,          // parámetro de no-centralidad
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

nu1

[in] Primer parámetro de distribución (número de grados de libertad).

nu2

[in] Segundo parámetro de distribución (número de grados de libertad).

sigma

[in] Parámetro de no-centralidad.

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

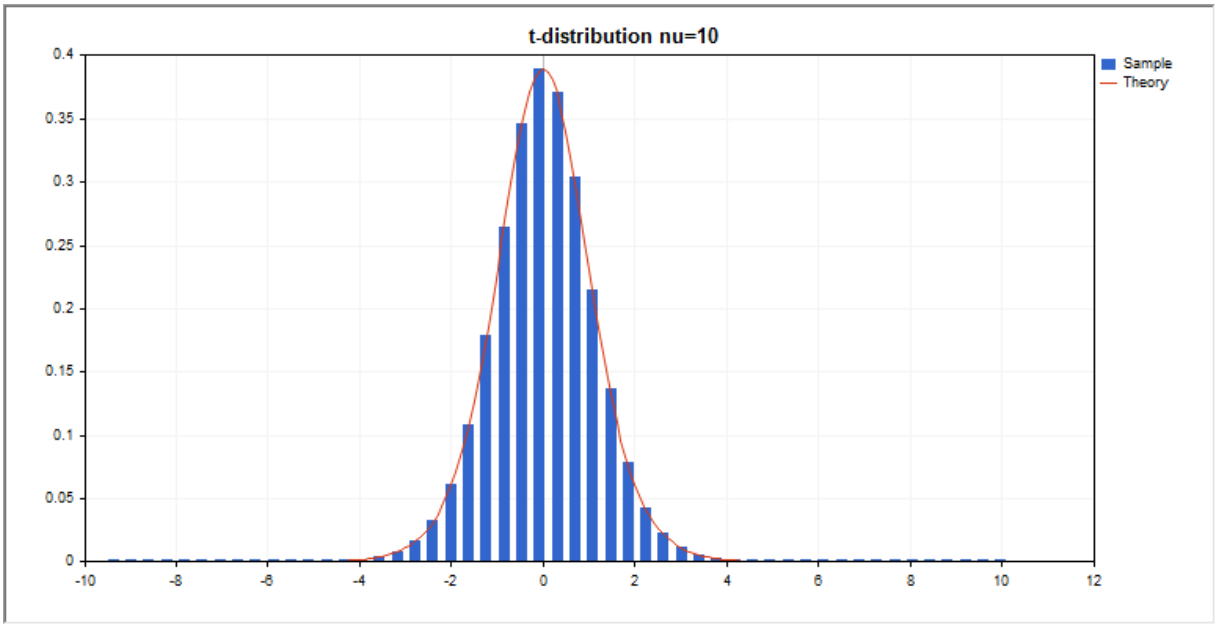
Distribución T

En este apartado se muestran las funciones para trabajar con la distribución T de Student. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley de Student. La distribución T se describe con la siguiente fórmula:

$$f_T(x|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\pi\nu}} \frac{1}{\left(1+\frac{x^2}{\nu}\right)^{\frac{\nu+1}{2}}}$$

donde:

- x — valor de la magnitud aleatoria
- ν — parámetro de distribución (número de grados de libertad)



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityT	Calcula la densidad de probabilidad de la distribución T
MathCumulativeDistributionT	Calcula el valor de la función de la distribución T de la probabilidad
MathQuantileT	Calcula el valor de la función inversa de la distribución T para la probabilidad indicada
MathRandomT	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de Student

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathMomentsT	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución de Student

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\T.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_par=10;     // número de grados de libertad
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // número de valores en la muestra
   int ncells=51;          // número de intervalos en el histograma
   double x[];             // centros de los intervalos del histograma
   double y[];             // número de valores de la muestra que han entrado en el inter
   double data[];          // muestra de valores aleatorios
   double max,min;         // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución T
   MathRandomT(nu_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityT(x2,nu_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("t-distribution nu=%G",nu_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityT

Calcula la densidad de la probabilidad de la distribución T de Student con el parámetro nu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityT(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución T de Student con el parámetro nu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityT(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución T de Student con el parámetro nu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dt() en R.

```
bool  MathProbabilityDensityT(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución T de Student con el parámetro nu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityT(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionT

Calcula la distribución de Student con el parámetro nu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionT(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de Student con el parámetro nu para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionT(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de Student con el parámetro nu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de pt() en R.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución de Student con el parámetro nu para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileT

Calcula el valor de la función inversa de la distribución T de Student con el parámetro nu para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileT(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución T de Student con el parámetro nu para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileT(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución T de Student con el parámetro nu para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qt() en R.

```
double  MathQuantileT(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución T de Student con el parámetro nu para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileT(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomT

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución T de Student con el parámetro nu. En caso de error, retorna NaN.

```
double  MathRandomT(
   const double  nu,              // parámetro de distribución (número de grados de lib
   int&          error_code       // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución T de Student con el parámetro nu. En caso de error, retorna false. Análogo de rt() en R.

```
bool  MathRandomT(
   const double  nu,              // parámetro de distribución (número de grados de lib
   const int     data_count,      // número de datos necesarios
   double&       result[]         // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

nu

[in] Parámetro de distribución (número de grados de libertad).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsT

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución T de Student con el parámetro nu.

```
double  MathMomentsT(
   const double  nu,              // parámetro de distribución (número de grados de lib
   double&       mean,            // variable para el valor medio
   double&       variance,        // variable para la varianza
   double&       skewness,        // variable para el coeficiente de asimetría
   double&       kurtosis,        // variable para la curtosis
   int&          error_code       // variable para el código de error
   );
```

Parámetros

nu

[in] Parámetro de distribución (número de grados de libertad).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

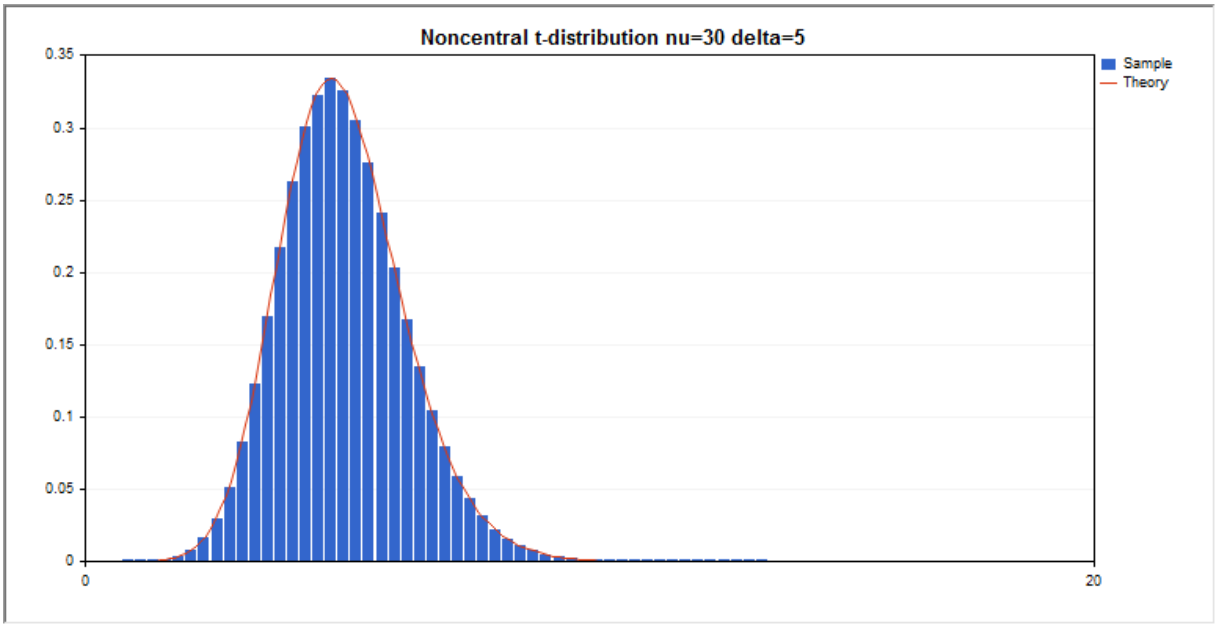
Distribución T

En este apartado se muestran las funciones para trabajar con la distribución T no central. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley de la distribución T no central. La distribución T no central se describe con la siguiente fórmula:

$$f_{NoncentralT}(x \mid \nu, \delta) = \frac{\nu^{\frac{\nu}{2}} e^{-\frac{\sigma^2}{2}}}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi}\left(\nu+x^2\right)^{\frac{\nu+1}{2}}} \sum_{r=0}^{\infty} \frac{(x\delta)^r}{r!} \left(\frac{2}{\nu+x^2}\right)^{\frac{r}{2}} \Gamma\left(\frac{\nu+r+1}{2}\right)$$

donde:

- x — valor de la magnitud aleatoria
- ν — parámetro de distribución (número de grados de libertad)
- σ — parámetro de no-centralidad



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityNoncentralT	Calcula la densidad de probabilidad de la distribución T no central
MathCumulativeDistributionNoncentralT	Calcula el valor de la función de la distribución T no central de la probabilidad
MathQuantileNoncentralT	Calcula el valor de la función inversa de la distribución T no central para la probabilidad indicada
MathRandomNoncentralT	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución Student no central

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathMomentsNoncentralT	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución de Student no central

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralT.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_par=30;      // número de grados de libertad
input double delta_par=5;    // parámetro de no-centralidad
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores en la muestra
   int ncells=51;       // número de intervalos en el histograma
   double x[];          // centros de los intervalos del histograma
   double y[];          // número de valores de la muestra que han entrado en el inter
   double data[];       // muestra de valores aleatorios
   double max,min;      // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución T no central
   MathRandomNoncentralT(nu_par,delta_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralT(x2,nu_par,delta_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral t-distribution nu=%G delta=%G",nu_p
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralT

Calcula la densidad de la probabilidad de la distribución T de Student no central con los parámetros nu y delta para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const double  delta,          // parámetro de no-centralidad
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución T de Student no central con los parámetros nu y delta para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const double  delta,          // parámetro de no-centralidad
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución T de Student no central con los parámetros nu y delta para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dt() en R.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  delta,          // parámetro de no-centralidad
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución T de Student no central con los parámetros nu y delta para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  delta,          // parámetro de no-centralidad
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

delta

[in] Parámetro de no-centralidad.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralT

Calcula la distribución de la probabilidad de la distribución T no central con los parámetros nu y delta para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const double  delta,          // parámetro de no-centralidad
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de la distribución T no central con los parámetros nu y delta para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // valor de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de libe
   const double  delta,          // parámetro de no-centralidad
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de la distribución T no central con los parámetros nu y delta para una magnitud aleatoria x. En caso de error, retorna false. Análogo de pt() en R.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  delta,          // parámetro de no-centralidad
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución de la probabilidad de la distribución T no central con los parámetros nu y delta para una magnitud aleatoria x. En caso de error, retorna false.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  delta,          // parámetro de no-centralidad
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

delta

[in] Parámetro de no-centralidad.

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNoncentralT

Calcula el valor de la función inversa de la distribución T de Student no central con los parámetros nu y delta para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileNoncentralT(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  delta,          // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución T de Student no central con los parámetros nu y delta para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileNoncentralT(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  delta,          // parámetro de no-centralidad
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución T de Student no central con los parámetros nu y delta para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qt() en R.

```
double  MathQuantileNoncentralT(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  delta,          // parámetro de no-centralidad
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución T de Student no central con los parámetros nu y delta para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileNoncentralT(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  delta,          // parámetro de no-centralidad
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

nu

[in] Parámetro de distribución (número de grados de libertad).

delta

[in] Parámetro de no-centralidad.

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralT

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución T de Student no central con los parámetros nu y delta. En caso de error, retorna NaN.

```
double  MathRandomNoncentralT(
   const double  nu,              // parámetro de distribución (número de grados de lib
   const double  delta,           // parámetro de no-centralidad
   int&          error_code       // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución T de Student no central con los parámetros nu y delta. En caso de error, retorna false. Análogo de rt() en R.

```
bool  MathRandomNoncentralT(
   const double  nu,              // parámetro de distribución (número de grados de lib
   const double  delta,           // parámetro de no-centralidad
   const int     data_count,      // número de datos necesarios
   double&       result[]         // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

nu

[in] Parámetro de distribución (número de grados de libertad).

delta

[in] Parámetro de no-centralidad.

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralT

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución T de Student no central con los parámetros nu y delta.

```
double  MathMomentsNoncentralT(
   const double  nu,             // parámetro de distribución (número de grados de lib
   const double  delta,          // parámetro de no-centralidad
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

nu

[in] Parámetro de distribución (número de grados de libertad).

delta

[in] Parámetro de no-centralidad.

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

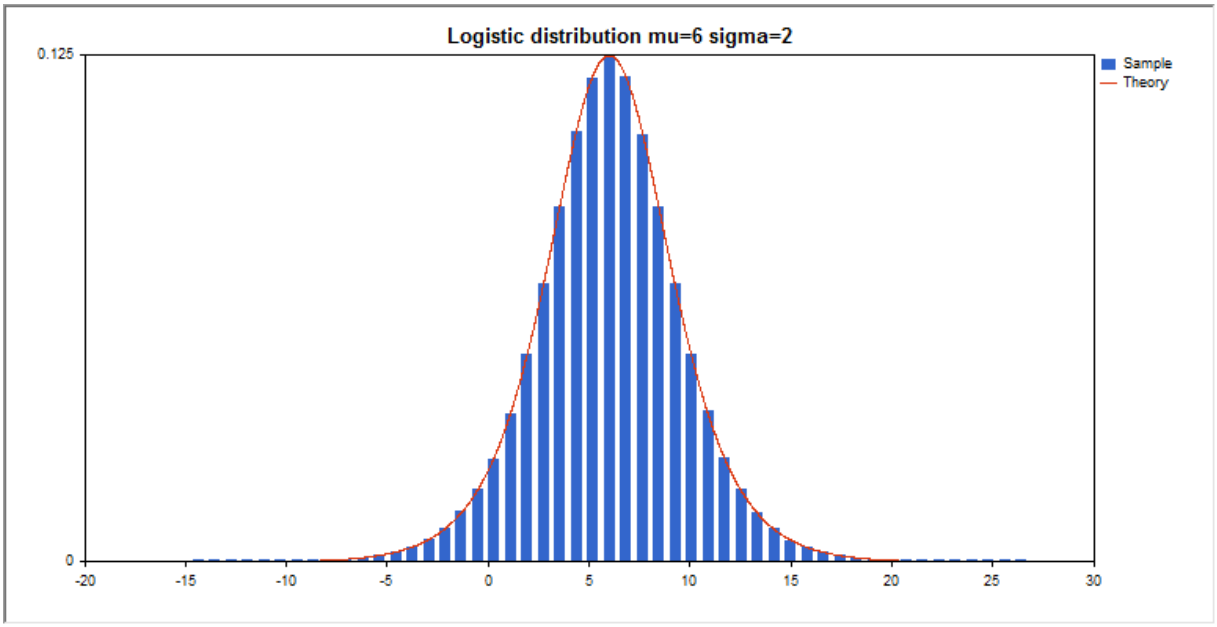
Distribución logística

En este apartado se muestran las funciones para trabajar con la distribución logística. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley logística. La distribución logística se describe con la siguiente fórmula:

$$f_{Logistic}(x|\mu,\sigma) = \frac{e^{\frac{x-\mu}{\sigma}}}{\sigma\left(1+e^{\frac{x-\mu}{\sigma}}\right)^2}$$

donde:

- x — valor de la magnitud aleatoria
- μ — parámetro de distribución mean
- σ — parámetro de distribución scale



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityLogistic	Calcula la densidad de probabilidad de la distribución logística
MathCumulativeDistributionLogistic	Calcula el valor de la función de la distribución logística de la probabilidad
MathQuantileLogistic	Calcula el valor de la función inversa de la distribución logística para la probabilidad indicada
MathRandomLogistic	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución logística

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathMomentsLogistic	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución logística.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Logistic.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double mu_par=6;         // parámetro de distribución mean
input double sigma_par=2;      // parámetro de distribución scale
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // número de valores en la muestra
   int ncells=51;        // número de intervalos en el histograma
   double x[];           // centros de los intervalos del histograma
   double y[];           // número de valores de la muestra que han entrado en el inter
   double data[];        // muestra de valores aleatorios
   double max,min;       // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución logística
   MathRandomLogistic(mu_par,sigma_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLogistic(x2,mu_par,sigma_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Logistic distribution mu=%G sigma=%G",mu_par,s
   graphic.BackgroundMainSize(16);
//--- desactivamos el escalado automático del eje Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

© 2000-2025, MetaQuotes Ltd.

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityLogistic

Calcula la densidad de la probabilidad de la distribución logística con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityLogistic(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución logística con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityLogistic(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución logística con los parámetros mu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dlogis() en R.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución logística con los parámetros mu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

mu

[in] Parámetro de distribución mean.

sigma

[in] Parámetro de distribución scale.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionLogistic

Calcula la distribución logística de la probabilidad con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución logística de la probabilidad con los parámetros mu y sigma para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // valor de la magnitud aleatoria
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución logística de la probabilidad con los parámetros mu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución logística de la probabilidad con los parámetros mu y sigma para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de plogis() en R.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

mu

[in] Parámetro de distribución mean.

sigma

[in] Parámetro de distribución scale.

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileLogistic

Calcula el valor de la función inversa de distribución logística con los parámetros mu y sigma para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileLogistic(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de distribución logística con los parámetros mu y sigma para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileLogistic(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de distribución logística con los parámetros mu y sigma para una matriz de valores de probabilidad *probability*. En caso de error, retorna false. Análogo de qlogis() en R.

```
double  MathQuantileLogistic(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de distribución logística con los parámetros mu y sigma para una matriz de valores de probabilidad *probability*. En caso de error, retorna false.

```
bool  MathQuantileLogistic(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

mu

[in] Parámetro de distribución mean.

sigma

[in] Parámetro de distribución scale.

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomLogistic

Genera una magnitud pseudoaleatoria, distribuida según la ley de distribución logística con los parámetros mu y sigma. En caso de error, retorna NaN.

```
double  MathRandomLogistic(
   const double  mu,              // parámetro de distribución mean
   const double  sigma,           // parámetro de distribución scale
   int&          error_code       // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de distribución logística con los parámetros mu y sigma. En caso de error, retorna false. Análogo de rlogis() en R.

```
bool  MathRandomLogistic(
   const double  mu,              // parámetro de distribución mean
   const double  sigma,           // parámetro de distribución scale
   const int     data_count,      // número de datos necesarios
   double&       result[]         // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

mu

[in] Parámetro de distribución mean.

sigma

[in] Parámetro de distribución scale.

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsLogistic

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución logística con los parámetros mu y sigma.

```
double  MathMomentsLogistic(
   const double  mu,             // parámetro de distribución mean
   const double  sigma,          // parámetro de distribución scale
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

mu

[in] Parámetro de distribución mean.

sigma

[in] Parámetro de distribución scale.

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

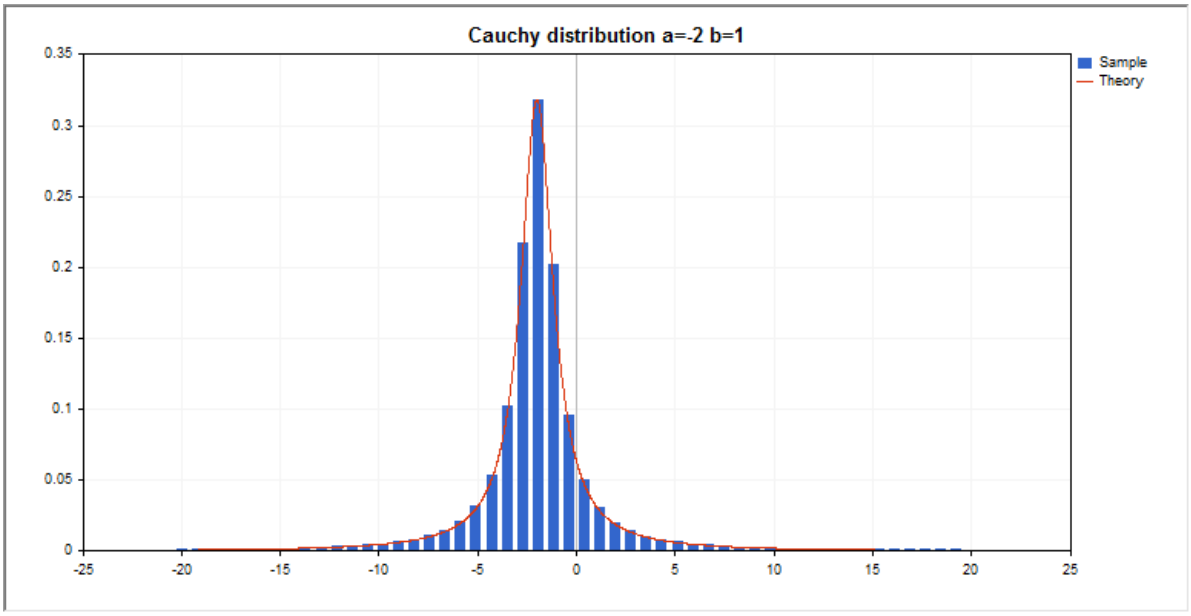
Distribución de Cauchy

En este apartado se muestran las funciones para trabajar con la distribución de Cauchy. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley de Cauchy. La distribución de Cauchy se describe con la siguiente fórmula:

$$f_{Cauchy}(x|a,b) = \frac{1}{\pi}\frac{b}{\left(b^2+(x-a)^2\right)}$$

donde:

- x — valor de la magnitud aleatoria
- a — parámetro de distribución mean
- b — parámetro de distribución scale



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityCauchy	Calcula la densidad de probabilidad de la distribución de Cauchy
MathCumulativeDistributionCauchy	Calcula el valor de la función de la distribución de la probabilidad de Cauchy
MathQuantileCauchy	Calcula el valor de la función inversa de la distribución de Cauchy para la probabilidad indicada
MathRandomCauchy	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de Cauchy
MathMomentsCauchy	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución de Cauchy

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Cauchy.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double a_par=-2;      // parámetro de distribución mean
input double b_par=1;       // parámetro de distribución scale
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // número de valores en la muestra
   int ncells=51;          // número de intervalos en el histograma
   double x[];             // centros de los intervalos del histograma
   double y[];             // número de valores de la muestra que han entrado en el inter
   double data[];          // muestra de valores aleatorios
   double max,min;         // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución de Cauchy
   MathRandomCauchy(a_par,b_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityCauchy(x2,a_par,b_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Cauchy distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1)
      return(false);
   int size=ArraySize(data);
   if(size<cells*10)
      return(false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   Print("min=",minv," max=",maxv);
   minv=-20;
   maxv=20;
   double range=maxv-minv;
   double width=range/cells;
   if(width==0)
      return(false);
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=(int)MathRound((data[i]-minv)/width);
      if(ind>=0 && ind<cells)
         frequency[ind]++;
     }
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityCauchy

Calcula la densidad de la probabilidad de la distribución de Cauchy con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityCauchy(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución de Cauchy con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityCauchy(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución de Cauchy con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dcauchy() en R.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución de Cauchy con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

a

[in] Parámetro de distribución mean.

b

[in] Parámetro de distribución scale.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionCauchy

Calcula la distribución de la probabilidad de Cauchy con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de Cauchy con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de la probabilidad de Cauchy con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución de la probabilidad de Cauchy con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de plogis() en R.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

a

[in] Parámetro de distribución mean.

b

[in] Parámetro de distribución scale.

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileCauchy

Calcula el valor de la función inversa de la distribución de Cauchy con los parámetros a y b para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileCauchy(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución de Cauchy con los parámetros a y b para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileCauchy(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución de Cauchy con los parámetros a y b para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qcauschy() en R.

```
double  MathQuantileCauchy(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución de Cauchy con los parámetros a y b para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileCauchy(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

a

[in] Parámetro de distribución mean.

b

[in] Parámetro de distribución scale.

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomCauchy

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución de Cauchy con los parámetros a y b. En caso de error, retorna NaN.

```
double  MathRandomCauchy(
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   int&          error_code      // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución de Cauchy con los parámetros a y b. En caso de error, retorna false. Análogo de rcauchy() en R.

```
bool  MathRandomCauchy(
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

a

[in] Parámetro de distribución mean.

b

[in] Parámetro de distribución scale.

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsCauchy

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución de Cauchy con los parámetros a y b.

```
double  MathMomentsCauchy(
   const double  a,              // parámetro de distribución mean
   const double  b,              // parámetro de distribución scale
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

a

[in] Parámetro de distribución mean.

b

[in] Parámetro de distribución scale.

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

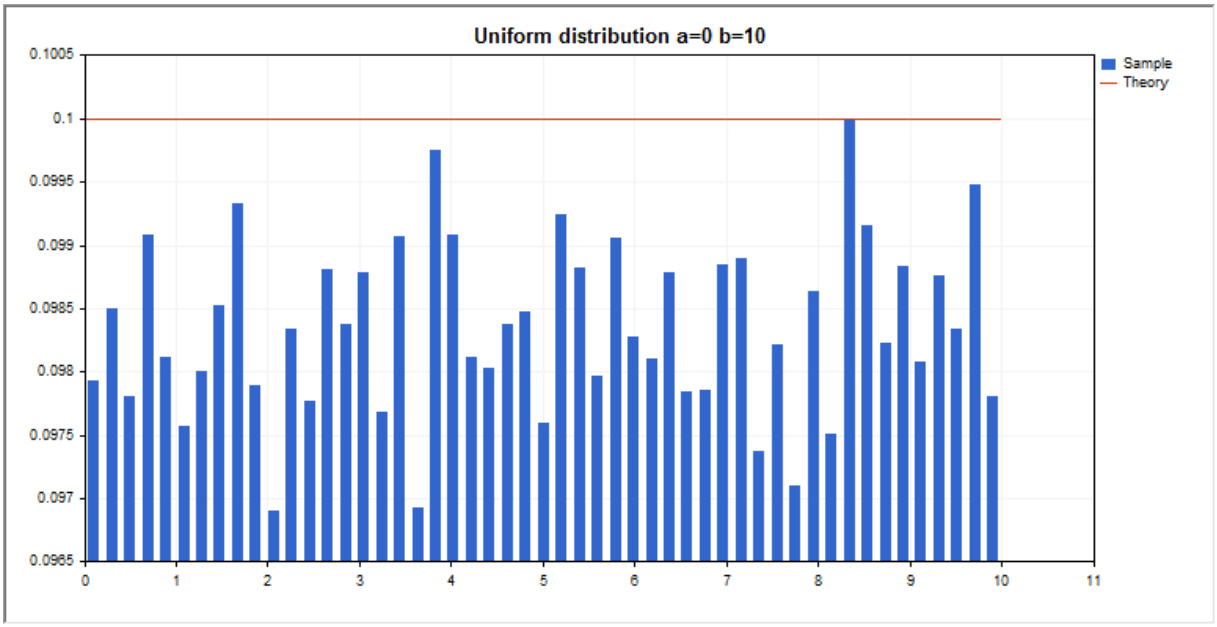
Distribución uniforme

En este apartado se muestran las funciones para trabajar con la distribución uniforme. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley uniforme. La distribución uniforme se describe con la siguiente fórmula:

$$f_{Uniform}(x|\,a,b) = \frac{1}{b-a}$$

donde:

- x — valor de la magnitud aleatoria
- a — parámetro de distribución (límite inferior)
- b — parámetro de distribución (límite superior)



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityUniform	Calcula la densidad de probabilidad de la distribución uniforme
MathCumulativeDistributionUniform	Calcula el valor de la función de la distribución uniforme de la probabilidad
MathQuantileUniform	Calcula el valor de la función inversa de la distribución uniforme para la probabilidad indicada
MathRandomUniform	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de distribución uniforme
MathMomentsUniform	Calcula los valores numéricos teóricos de los 4 primeros momentos para la distribución uniforme.

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Uniform.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double a_par=0;      // parámetro de distribución a (límite inferior)
input double b_par=10;     // parámetro de distribución b (límite superior)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores en la muestra
   int ncells=51;       // número de intervalos en el histograma
   double x[];          // centros de los intervalos del histograma
   double y[];          // número de valores de la muestra que han entrado en el inter
   double data[];       // muestra de valores aleatorios
   double max,min;      // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución uniforme
   MathRandomUniform(a_par,b_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityUniform(x2,a_par,b_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Uniform distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityUniform

Calcula la densidad de la probabilidad de la distribución uniforme con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityUniform(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución uniforme con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityUniform(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución uniforme con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dunif() en R.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución uniforme con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

a

[in] Parámetro de distribución a (límite inferior).

b

[in] Parámetro de distribución b (límite superior).

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionUniform

Calcula la distribución uniforme de la probabilidad con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionUniform(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución uniforme de la probabilidad con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionUniform(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución uniforme de la probabilidad con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución uniforme de la probabilidad con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de punif() en R.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

a

[in] Parámetro de distribución a (límite inferior).

b

[in] Parámetro de distribución b (límite superior).

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileUniform

Calcula el valor de la función inversa de la distribución uniforme con los parámetros a y b para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileUniform(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución uniforme con los parámetros a y b para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileUniform(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución uniforme con los parámetros a y b para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qcauschy() en R.

```
double  MathQuantileUniform(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución uniforme con los parámetros a y b para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileUniform(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

a

[in] Parámetro de distribución a (límite inferior).

b

[in] Parámetro de distribución b (límite superior).

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomUniform

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución uniforme con los parámetros a y b. En caso de error, retorna NaN.

```
double  MathRandomUniform(
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   int&          error_code      // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución uniforme con los parámetros a y b. En caso de error, retorna false. Análogo de runif() en R.

```
bool  MathRandomUniform(
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

a

[in] Parámetro de distribución a (límite inferior).

b

[in] Parámetro de distribución b (límite superior).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsUniform

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución uniforme con los parámetros a y b.

```
double  MathMomentsUniform(
   const double  a,              // parámetro de distribución a (límite inferior)
   const double  b,              // parámetro de distribución b (límite superior)
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

a

[in] Parámetro de distribución a (límite inferior).

b

[in] Parámetro de distribución b (límite superior).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

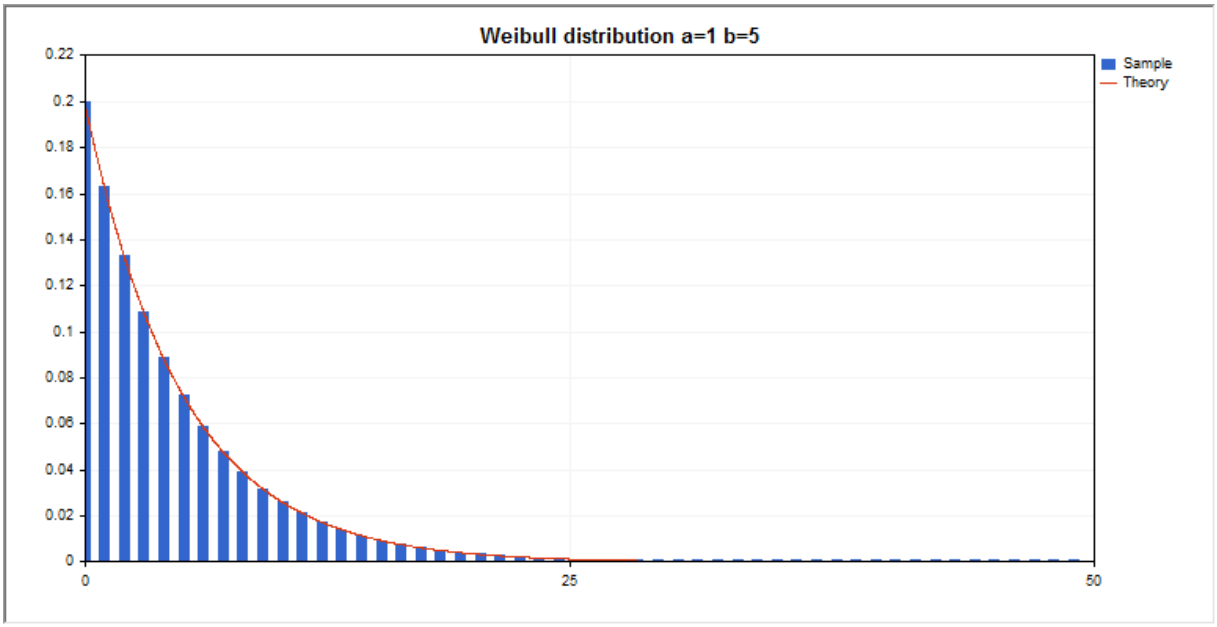
Distribución de Weibull

En este apartado se muestran las funciones para trabajar con la distribución de Weibull. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley de Weibull. La distribución de Weibull se describe con la siguiente fórmula:

$$f_{Weibull}(x|a,b) = \frac{a}{b}\left(\frac{x}{b}\right)^{a-1} e^{-\left(\frac{x}{b}\right)^{a}}$$

donde:

- x — valor de la magnitud aleatoria
- a — parámetro de distribución (shape)
- b — parámetro de distribución (scale)



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityWeibull	Calcula la densidad de probabilidad de la distribución de Weibull
MathCumulativeDistributionWeibull	Calcula el valor de la función de la distribución de la probabilidad de Weibull
MathQuantileWeibull	Calcula el valor de la función inversa de la distribución de Weibull para la probabilidad indicada
MathRandomWeibull	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de Weibull

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathMomentsWeibull	La función calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución de Weibull.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Weibull.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double a_par=1;      // parámetro de distribución (shape)
input double b_par=5;      // parámetro de distribución (scale)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores en la muestra
   int ncells=51;       // número de intervalos en el histograma
   double x[];          // centros de los intervalos del histograma
   double y[];          // número de valores de la muestra que han entrado en el inter
   double data[];       // muestra de valores aleatorios
   double max,min;      // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución de Weibull
   MathRandomWeibull(a_par,b_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityWeibull(x2,a_par,b_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Weibull distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- desactivamos el escalado automático del eje X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

© 2000-2025, MetaQuotes Ltd.

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityWeibull

Calcula la densidad de la probabilidad de la distribución de Weibull con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución de Weibull con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   int&          error_code      // variable para el código de error
   );
```

Calcula la densidad de la probabilidad de la distribución de Weibull con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dweibull() en R.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula la densidad de la probabilidad de la distribución de Weibull con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

a

[in] Parámetro de distribución (scale).

b

[in] Parámetro de distribución (shape).

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionWeibull

Calcula la distribución de Weibull con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de Weibull con los parámetros a y b para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // valor de la magnitud aleatoria
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   int&          error_code      // variable para el código de error
   );
```

Calcula la distribución de Weibull con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dnbinom() en R.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la distribución de Weibull con los parámetros a y b para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

a

[in] Parámetro de distribución (scale).

b

[in] Parámetro de distribución (shape).

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileWeibull

Calcula el valor de la función inversa de la distribución de probabilidad de Weibull con los parámetros a y b para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileWeibull(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución de probabilidad de Weibull con los parámetros a y b para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileWeibull(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor de la función inversa de la distribución de probabilidad de Weibull con los parámetros a y b para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qweibull() en R.

```
double  MathQuantileWeibull(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor de la función inversa de la distribución de probabilidad de Weibull con los parámetros a y b para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileWeibull(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

a

[in] Parámetro de distribución (scale).

b

[in] Parámetro de distribución (shape).

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomWeibull

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución de Weibull con los parámetros a y b. En caso de error, retorna NaN.

```
double  MathRandomWeibull(
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   int&          error_code      // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución de Weibull con los parámetros a y b. En caso de error, retorna false. Análogo de rnbinom() en R.

```
bool  MathRandomWeibull(
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

a

[in] Parámetro de distribución (scale).

b

[in] Parámetro de distribución (shape).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsWeibull

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución de Weibull con los parámetros a y b.

```
double  MathMomentsWeibull(
   const double  a,              // parámetro de la distribución (shape)
   const double  b,              // parámetro de la distribución (scale)
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

a

[in] Parámetro de distribución (scale).

b

[in] Parámetro de distribución (shape).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

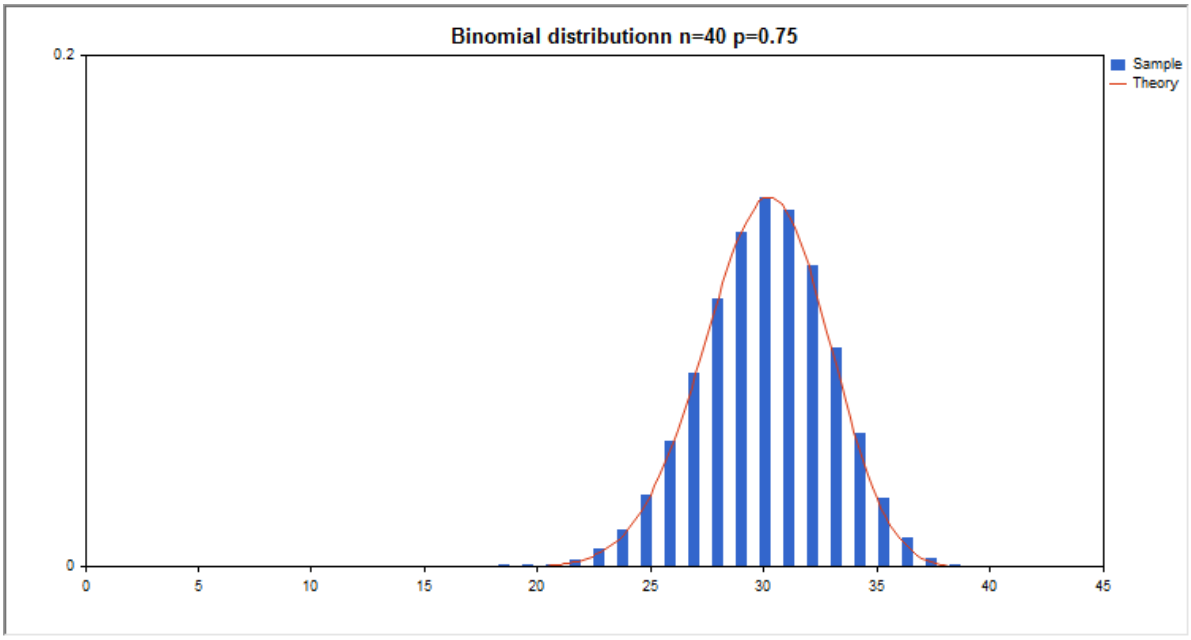
Distribución binomial

En este apartado se muestran las funciones para trabajar con la distribución binomial. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley binomial. La distribución binomial se describe con la siguiente fórmula:

$$f_{Binomial}(x \mid n, p) = \binom{n}{x} p^x (1-p)^{n-x}$$

donde:

- x — valor de la magnitud aleatoria
- n — número de pruebas
- p — probabilidad de éxito para cada prueba



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityBinomial	Calcula la densidad de probabilidad de la distribución binomial
MathCumulativeDistributionBinomial	Calcula el valor de la función de la distribución binomial de la probabilidad
MathQuantileBinomial	Calcula el valor de la función inversa de la distribución binomial para la probabilidad indicada
MathRandomBinomial	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución binomial
MathMomentsBinomial	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución binomial.

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Binomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double n_par=40;         // número de pruebas
input double p_par=0.75;       // probabilidad de éxito para cada prueba
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // número de valores en la muestra
   int ncells=20;          // número de intervalos en el histograma
   double x[];             // centros de los intervalos del histograma
   double y[];             // número de valores de la muestra que han entrado en el inter
   double data[];          // muestra de valores aleatorios
   double max,min;         // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución binomial
   MathRandomBinomial(n_par,p_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityBinomial(x2,n_par,p_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Binomial distributionn n=%G p=%G",n_par,p_par)
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityBinomial

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución binomial con los parámetros n y p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // valor de la magnitud aleatoria
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito pa
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución binomial con los parámetros n y p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // valor de la magnitud aleatoria
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito pa
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución binomial con los parámetros n y p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dbinom() en R.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución binomial con los parámetros n y p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

n

[in] Parámetro de distribución (número de pruebas).

p

[in] Parámetro de distribución (probabilidad de éxito para cada prueba).

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionBinomial

Calcula la función de distribución para la ley binomial negativa con los parámetros n y p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // valor de la magnitud aleatoria
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito pa
   const bool    tail,           // bandera para calcular, si true, entonces se calcula
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula la función de distribución para la ley binomial negativa con los parámetros n y p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // valor de la magnitud aleatoria
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito pa
   int&          error_code      // variable para el código de error
   );
```

Calcula la función de distribución para la ley binomial negativa con los parámetros n y p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dnbinom() en R.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   const bool    tail,           // bandera para calcular, si true, entonces se calcul
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la función de distribución para la ley binomial negativa con los parámetros n y p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

n

[in] Parámetro de distribución (número de pruebas).

p

[in] Parámetro de distribución (probabilidad de éxito para cada prueba).

tail

[in] Bandera de cálculo. Si true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileBinomial

Calcula el valor inverso de la función de distribución para la ley binomial con los parámetros n y p para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileBinomial(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor inverso de la función de distribución para la ley binomial con los parámetros n y p para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileBinomial(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor inverso de la función de distribución para la ley binomial con los parámetros n y p para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qbinom() en R.

```
double  MathQuantileBinomial(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor inverso de la función de distribución para la ley binomial con los parámetros n y p para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileBinomial(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

n

[in] Parámetro de distribución (número de pruebas).

p

[in] Parámetro de distribución (probabilidad de éxito para cada prueba).

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomBinomial

Genera una magnitud pseudoaleatoria, distribuida según la ley de distribución binominal negativa con los parámetros n y p. En caso de error, retorna NaN.

```
double  MathRandomBinomial(
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   int&          error_code      // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de distribución binominal negativa con los parámetros n y p. En caso de error, retorna false. Análogo de rnbinom() en R.

```
bool  MathRandomBinomial(
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

n

[in] Parámetro de distribución (número de pruebas).

p

[in] Parámetro de distribución (probabilidad de éxito para cada prueba).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsBinomial

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución binomial con los parámetros n y p.

```
double  MathMomentsBinomial(
   const double  n,              // parámetro de distribución (número de pruebas)
   const double  p,              // parámetro de distribución (probabilidad de éxito p
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

n

[in] Parámetro de distribución (número de pruebas).

p

[in] Parámetro de distribución (probabilidad de éxito para cada prueba).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

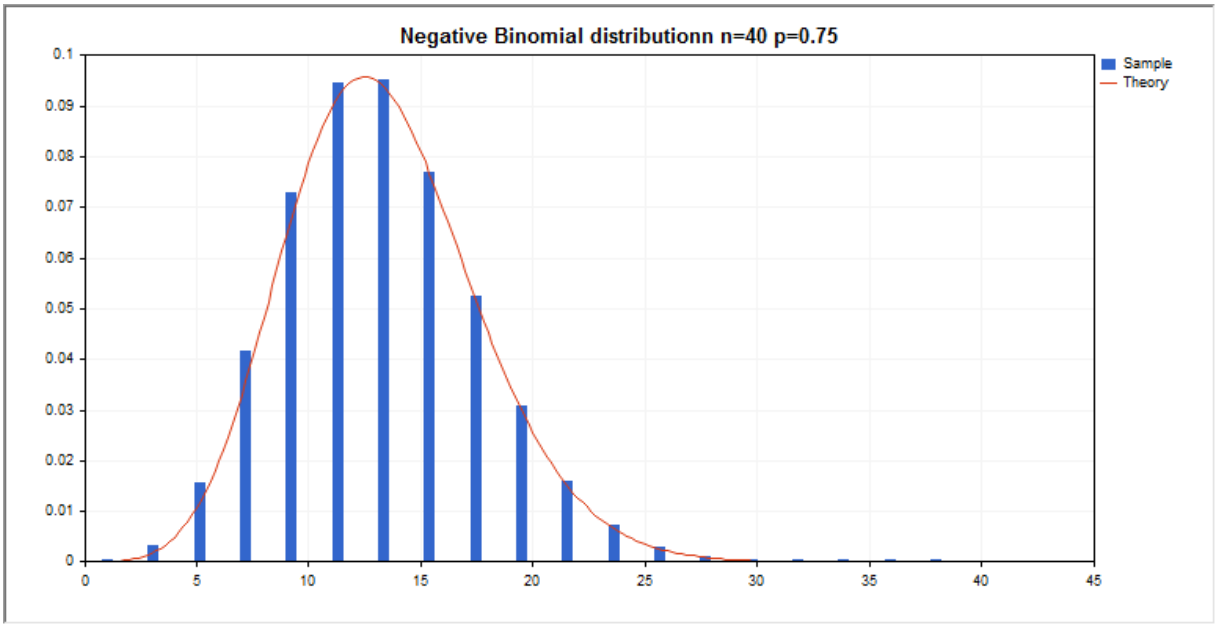
Distribución binomial negativa

En este apartado se muestran las funciones para trabajar con la distribución binomial negativa. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley binomial negativa. La distribución binomial negativa se describe con la siguiente fórmula:

$$f_{NegativeBinomial}(x \mid r, p) = \frac{\Gamma(r+x)}{\Gamma(r)\Gamma(x+1)} p^r (1-p)^x$$

donde:

- x — valor de la magnitud aleatoria
- r — número de pruebas exitosas
- p — probabilidad de éxito



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityNegativeBinomial	Calcula la densidad de probabilidad de la distribución binomial negativa
MathCumulativeDistributionNegativeBinomial	Calcula el valor de la función de la distribución binomial negativa de la probabilidad
MathQuantileNegativeBinomial	Calcula el valor de la función inversa de la distribución binomial negativa para la probabilidad indicada
MathRandomNegativeBinomial	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución binomial negativa

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathMomentsNegativeBinomial	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución binomial negativa.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NegativeBinomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double n_par=40;         // número de pruebas
input double p_par=0.75;       // probabilidad de éxito para cada prueba
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // número de valores en la muestra
   int ncells=19;        // número de intervalos en el histograma
   double x[];           // centros de los intervalos del histograma
   double y[];           // número de valores de la muestra que han entrado en el inter
   double data[];        // muestra de valores aleatorios
   double max,min;       // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución binomial negativa
   MathRandomNegativeBinomial(n_par,p_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityNegativeBinomial(x2,n_par,p_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
```

© 2000-2025, MetaQuotes Ltd.

```
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Negative Binomial distributionn n=%G p=%G",n_p
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNegativeBinomial

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución binomial negativa con los parámetros r y p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución binomial negativa con los parámetros r y p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución binomial negativa con los parámetros r y p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dnbinom() en R.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución binomial negativa con los parámetros r y p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

© 2000-2025, MetaQuotes Ltd.

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

r

[in] Número de pruebas exitosas

p

[in] Probabilidad de éxito.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNegativeBinomial

Calcula la función de distribución para la ley binomial negativa con los parámetros r y p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula la función de distribución para la ley binomial negativa con los parámetros r y p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   int&          error_code      // variable para el código de error
   );
```

Calcula la función de distribución para la ley binomial negativa con los parámetros r y p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dnbinom() en R.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Calcula la función de distribución para la ley binomial negativa con los parámetros r y p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   double&       result[]        // matriz para los valores de la función de probabili
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Matriz con los valores de la magnitud aleatoria.

r

[in] Número de pruebas exitosas.

p

[in] Probabilidad de éxito.

tail

[in] Bandera para calcular, si lower_tail=true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor, si log_mode=true, entonces se calcula el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNegativeBinomial

Calcula el valor inverso de la función de distribución para la ley binomial negativa con los parámetros r y p para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor inverso de la función de distribución para la ley binomial negativa con los parámetros r y p para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor inverso de la función de distribución para la ley binomial negativa con los parámetros r y p para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qnbinom() en R.

```
double  MathQuantileNegativeBinomial(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor inverso de la función de distribución para la ley binomial negativa con los parámetros r y p para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileNegativeBinomial(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

r

[in] Número de pruebas exitosas.

p

[in] Probabilidad de éxito.

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomNegativeBinomial

Genera una magnitud pseudoaleatoria, distribuida según la ley de distribución binominal negativa con los parámetros r y p. En caso de error, retorna NaN.

```
double  MathRandomNegativeBinomial(
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   int&          error_code      // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de distribución binominal negativa con los parámetros r y p. En caso de error, retorna false. Análogo de rnbinom() en R.

```
bool  MathRandomNegativeBinomial(
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

r

[in] Número de pruebas exitosas.

p

[in] Probabilidad de éxito.

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNegativeBinomial

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución binomial negativa con los parámetros r y p.

```
double  MathMomentsNegativeBinomial(
   const double  r,              // número de pruebas exitosas
   const double  p,              // probabilidad de éxito
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

r

[in] Número de pruebas exitosas.

p

[in] Probabilidad de éxito.

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

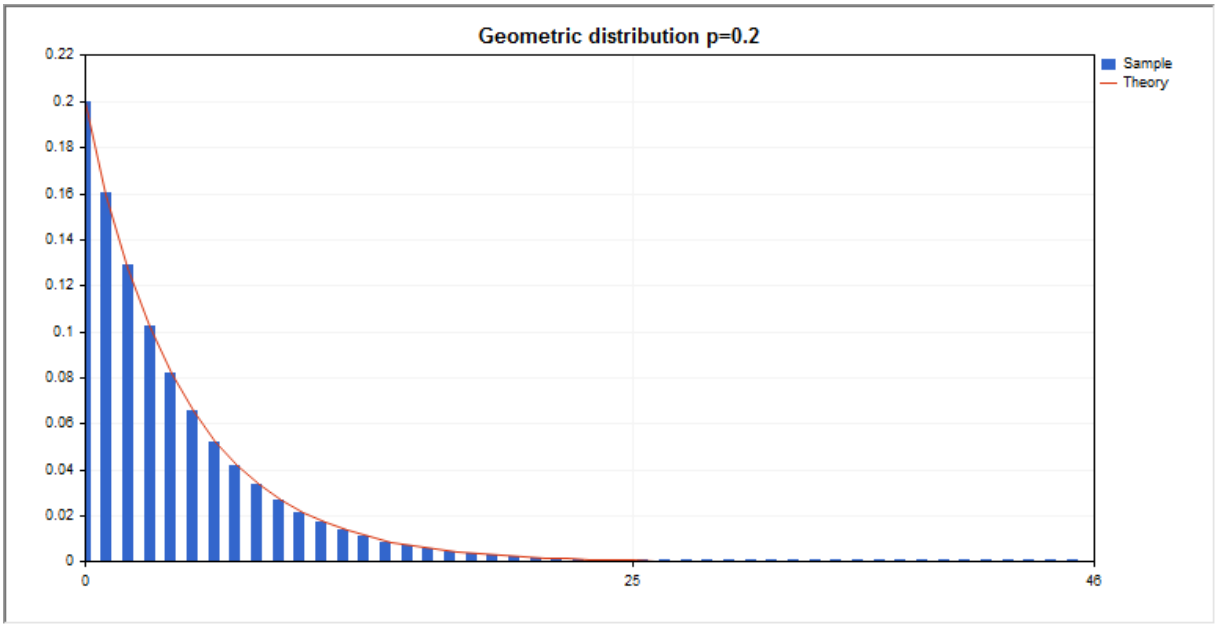
Distribución geométrica

En este apartado se muestran las funciones para trabajar con la distribución geométrica. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley geométrica. La distribución geométrica se describe con la siguiente fórmula:

$$f_{Geometric}(x|p) = p(1-p)^x$$

donde:

- x — valor de la magnitud aleatoria (número entero)
- p — probabilidad de aparición de un evento en una prueba



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityGeometric	Calcula la densidad de probabilidad de la distribución geométrica
MathCumulativeDistributionGeometric	Calcula el valor de la función de la distribución geométrica de la probabilidad
MathQuantileGeometric	Calcula el valor de la función inversa de la distribución geométrica para la probabilidad indicada
MathRandomGeometric	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución geométrica
MathMomentsGeometric	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución geométrica.

Ejemplo:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Geometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double p_par=0.2;      // probabilidad de aparición de un evento en una prueba
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores en la muestra
   int ncells=47;       // número de intervalos en el histograma
   double x[];          // centros de los intervalos del histograma
   double y[];          // número de valores de la muestra que han entrado en el inter
   double data[];       // muestra de valores aleatorios
   double max,min;      // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución  geométrica
   MathRandomGeometric(p_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(0,ncells,1,x2);
   MathProbabilityDensityGeometric(x2,p_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMain(StringFormat("Geometric distribution p=%G",p_par));
   graphic.BackgroundMainSize(16);
//--- desactivamos el escalado automático del eje X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityGeometric

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución geométrica con el parámetro p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  p,              // parámetro de distribución (probabilidad de aparici
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución geométrica con el parámetro p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  p,              // parámetro de distribución (probabilidad de aparici
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución geométrica con el parámetro p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dgeom() en R.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  p,              // parámetro de distribución (probabilidad de aparici
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución geométrica con el parámetro p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  p,              // parámetro de distribución (probabilidad de aparici
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

p

[in] Parámetro de distribución (probabilidad de aparición de un evento en una prueba).

© 2000-2025, MetaQuotes Ltd.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionGeometric

Calcula la función de distribución para la ley geométrica con el parámetro p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  p,              // parámetro de distribución (probabilidad de aparici
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula la función de distribución para la ley geométrica con el parámetro p para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  p,              // parámetro de distribución (probabilidad de aparici
   int&          error_code      // variable para el código de error
   );
```

Calcula la función de distribución para la ley geométrica con el parámetro p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de pgeom() en R.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  p,              // parámetro de distribución (probabilidad de aparic
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de probabil
   );
```

Calcula la función de distribución para la ley geométrica con el parámetro p para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  p,              // parámetro de distribución (probabilidad de aparic
   double&       result[]        // matriz para los valores de la función de probabil
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

p

[in] Parámetro de distribución (probabilidad de aparición de un evento en una prueba).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Bandera para calcular, si tail=true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor, si log_mode=true, entonces se calcula el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de probabilidad.

© 2000-2025, MetaQuotes Ltd.

MathQuantileGeometric

Calcula el valor inverso de la función de distribución para la ley geométrica con el parámetro p para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileGeometric(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  p,              // parámetro de distribución (probabilidad de aparici
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor inverso de la función de distribución para la ley geométrica con el parámetro p para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileGeometric(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  p,              // parámetro de distribución (probabilidad de aparici
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor inverso de la función de distribución para la ley geométrica con el parámetro p para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false. Análogo de qgeom() en R.

```
double  MathQuantileGeometric(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  p,              // parámetro de distribución (probabilidad de aparici
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor inverso de la función de distribución para la ley geométrica con el parámetro p para una matriz de valores de probabilidad *probability[]*. En caso de error, retorna false.

```
bool  MathQuantileGeometric(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  p,              // parámetro de distribución (probabilidad de aparici
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

p

© 2000-2025, MetaQuotes Ltd.

[in] Parámetro de distribución (probabilidad de aparición de un evento en una prueba).

tail

[in] Bandera para calcular, si false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomGeometric

Genera una magnitud pseudoaleatoria, distribuida según la ley de distribución geométrica con el parámetro p. En caso de error, retorna NaN.

```
double  MathRandomGeometric(
   const double  p,              // parámetro de distribución (probabilidad de aparici
   int&          error_code      // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de distribución geométrica con el parámetro p. En caso de error, retorna false. Análogo de rgeom() en R.

```
bool  MathRandomGeometric(
   const double  p,              // parámetro de distribución (probabilidad de aparici
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

p

[in] Parámetro de distribución (probabilidad de aparición de un evento en una prueba).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsGeometric

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución geométrica con el parámetro p.

```
double  MathMomentsGeometric(
   const double  p,              // parámetro de distribución (probabilidad de aparici
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

p

[in] Parámetro de distribución (probabilidad de aparición de un evento en una prueba).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

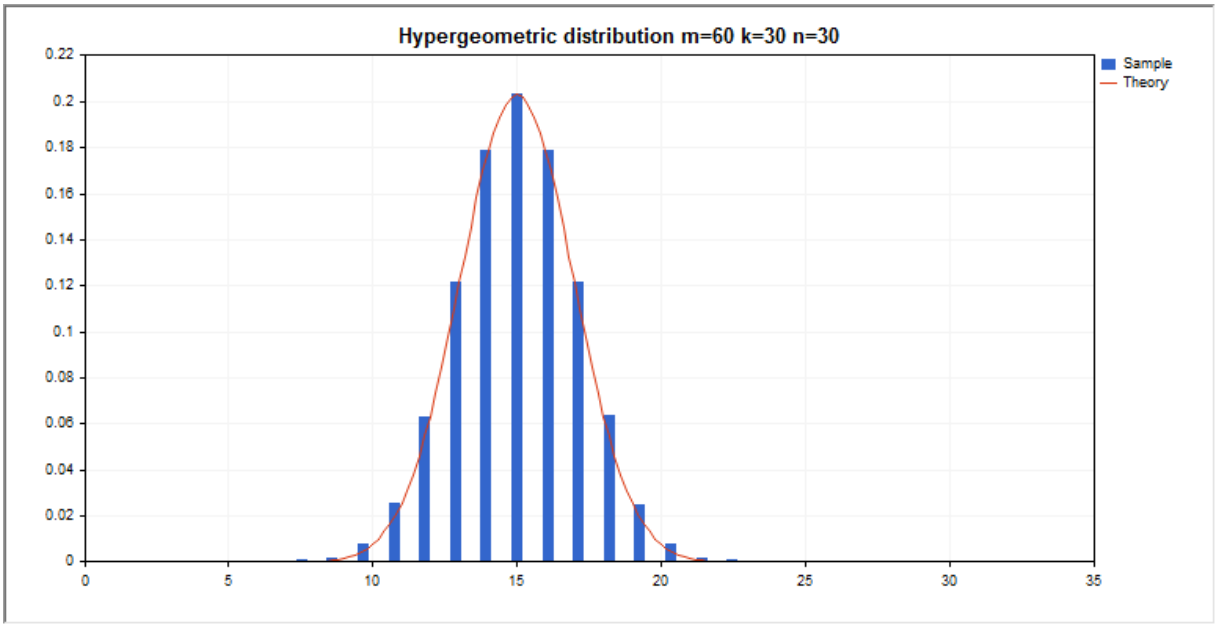
Distribución hipergeométrica

En este apartado se muestran las funciones para trabajar con la distribución hipergeométrica. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley hipergeométrica. La distribución hipergeométrica se describe con la siguiente fórmula:

$$f_{Hypergeometric}(x \mid m, k, n) = \frac{\binom{k}{x}\binom{m-k}{n-x}}{\binom{m}{n}}$$

donde:

- x — valor de la magnitud aleatoria (número entero)
- m — número total de objetos
- k — número de objetos con la característica deseada
- n — número de objetos tomados



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityHypergeometric	Calcula la densidad de probabilidad de la distribución hipergeométrica
MathCumulativeDistributionHypergeometric	Calcula el valor de la función de la distribución hipergeométrica de la probabilidad
MathQuantileHypergeometric	Calcula el valor de la función inversa de la distribución hipergeométrica para la probabilidad indicada

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathRandomHypergeometric	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de la distribución hipergeométrica
MathMomentsHypergeometric	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución hipergeométrica.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Hypergeometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double m_par=60;      // número total de objetos
input double k_par=30;      // número de objetos con la característica deseada
input double n_par=30;      // número de objetos tomados
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores en la muestra
   int ncells=15;       // número de intervalos en el histograma
   double x[];          // centros de los intervalos del histograma
   double y[];          // número de valores de la muestra que han entrado en el inter
   double data[];       // muestra de valores aleatorios
   double max,min;      // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución  hipergeométrica
   MathRandomHypergeometric(m_par,k_par,n_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
```

© 2000-2025, MetaQuotes Ltd.

```
   MathProbabilityDensityHypergeometric(x2,m_par,k_par,n_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Hypergeometric distribution m=%G k=%G n=%G",m_
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
```

© 2000-2025, MetaQuotes Ltd.

```
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityHypergeometric

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución hipergeométrica con los parámetros m, k y n para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (nú
   const double  n,              // número de objetos tomados (número entero)
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución hipergeométrica con los parámetros m, k y n para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (nú
   const double  n,              // número de objetos tomados (número entero)
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución hipergeométrica con los parámetros m, k y n para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dhyper() en R.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución hipergeométrica con los parámetros m, k y n para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   double&       result[]        // matriz para los valores de la función de densidad
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

m

[in] Número total de objetos (número entero).

k

[in] Número de objetos con la característica deseada (número entero).

n

[in] Número de objetos tomados (número entero).

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionHypergeometric

Calcula el valor de la función de distribución para la ley hipergeométrica con los parámetros m, k y n para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (nú
   const double  n,              // número de objetos tomados (número entero)
   const double  tail,           // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si lo
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de distribución para la ley hipergeométrica con los parámetros m, k y n para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (nú
   const double  n,              // número de objetos tomados (número entero)
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de distribución para la ley hipergeométrica con los parámetros m, k y n para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dhyper() en R.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   const double  tail,           // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si l
   double&       result[]        // matriz para los valores de la función de distribuc
   );
```

Calcula el valor de la función de distribución para la ley hipergeométrica con los parámetros m, k y n para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   double&       result[]        // matriz para los valores de la función de distribuc
   );
```

© 2000-2025, MetaQuotes Ltd.

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

m

[in] Número total de objetos (número entero).

k

[in] Número de objetos con la característica deseada (número entero).

n

[in] Número de objetos tomados (número entero).

tail

[in] Bandera para calcular, si lower_tail=true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor, si log_mode=true, entonces se calcula el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de distribución.

© 2000-2025, MetaQuotes Ltd.

MathQuantileHypergeometric

Calcula el valor inverso de la función de distribución para la ley hipergeométrica con los parámetros m, k y n para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileHypergeometric(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor inverso de la función de distribución para la ley hipergeométrica con los parámetros m, k y n para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantileHypergeometric(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor inverso de la función de distribución para la ley hipergeométrica con los parámetros m, k y n para una matriz de valores de probabilidad *probability*. En caso de error, retorna false. Análogo de qhyper() en R.

```
double  MathQuantileHypergeometric(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor inverso de la función de distribución para la ley hipergeométrica con los parámetros m, k y n para una matriz de valores de probabilidad *probability*. En caso de error, retorna false.

```
bool  MathQuantileHypergeometric(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

© 2000-2025, MetaQuotes Ltd.

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

m

[in] Número total de objetos (número entero).

k

[in] Número de objetos con la característica deseada (número entero).

n

[in] Número de objetos tomados (número entero).

tail

[in] bandera para calcular, si tail=false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomHypergeometric

Genera una magnitud pseudoaleatoria, distribuida según la ley de distribución hipergeométrica con los parámetros m, n y k. En caso de error, retorna NaN.

```
double  MathRandomHypergeometric(
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   int&          error_code      // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de distribución hipergeométrica con los parámetros m, n y k. En caso de error, retorna false. Análogo de rgeom() en R.

```
bool  MathRandomHypergeometric(
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   const int     data_count,     // número de datos necesarios
   double&       result[]        // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

m

[in] Número total de objetos (número entero).

k

[in] Número de objetos con la característica deseada (número entero).

n

[in] Número de objetos tomados (número entero).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsHypergeometric

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución hipergeométrica con los parámetros m, n y k.

```
double  MathMomentsHypergeometric(
   const double  m,              // número total de objetos (número entero)
   const double  k,              // número de objetos con la característica deseada (n
   const double  n,              // número de objetos tomados (número entero)
   double&       mean,           // variable para el valor medio
   double&       variance,       // variable para la varianza
   double&       skewness,       // variable para el coeficiente de asimetría
   double&       kurtosis,       // variable para la curtosis
   int&          error_code      // variable para el código de error
   );
```

Parámetros

m

[in] Número total de objetos (número entero).

k

[in] Número de objetos con la característica deseada (número entero).

n

[in] Número de objetos tomados (número entero).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

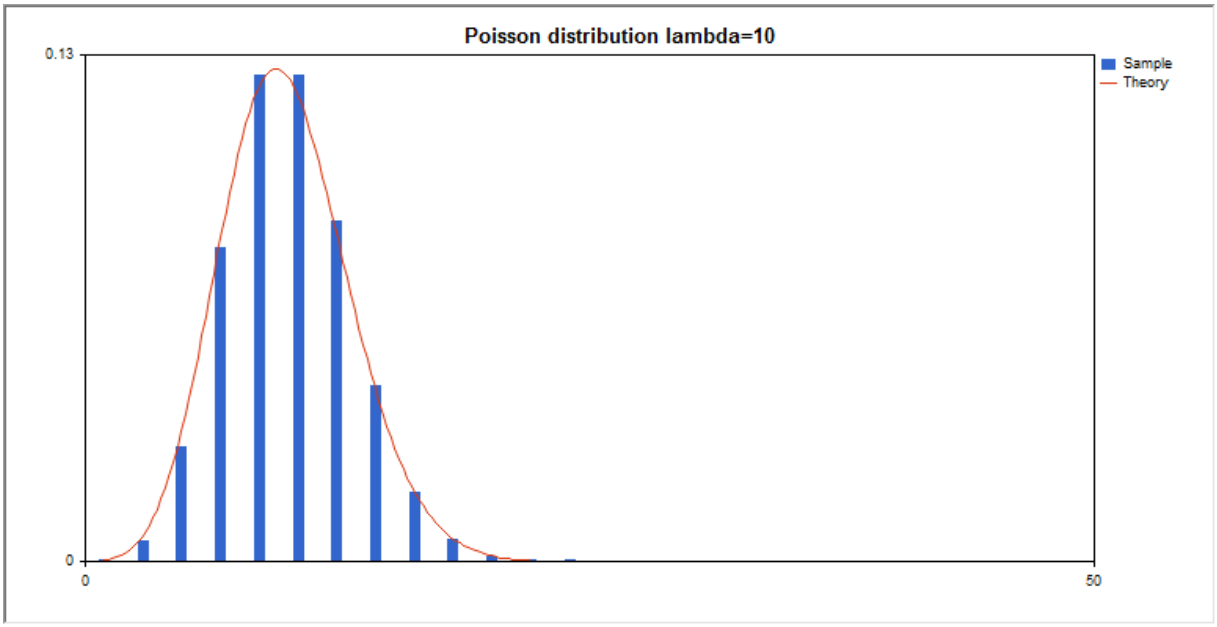
Distribución de Poisson

En este apartado se muestran las funciones para trabajar con la distribución de Poisson. Estas permiten calcular la densidad, la probabilidad, los cuantiles y generar números pseudoaleatorios, distribuidos conforme a la ley de Poisson. La distribución de Poisson se describe con la siguiente fórmula:

$$f_{Poisson}(x|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}$$

donde:

- x — valor de la magnitud aleatoria
- λ — parámetro de distribución (mean)



Aparte del cálculo de diferentes magnitudes aleatorias, se ha implementado la posibilidad de trabajar con sus matrices.

Función	Descripción
MathProbabilityDensityPoisson	Calcula la densidad de probabilidad de la distribución de Poisson
MathCumulativeDistributionPoisson	Calcula el valor de la función de la distribución de la probabilidad de Poisson
MathQuantilePoisson	Calcula el valor de la función inversa de la distribución de Poisson para la probabilidad indicada
MathRandomPoisson	Genera una magnitud/matriz pseudoaleatoria de magnitudes pseudoaleatorias, distribuidas según la ley de Poisson
MathMomentsPoisson	Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución de Poisson

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Poisson.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double lambda_par=10;      // parámetro de distribución (mean)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desactivamos la exhibición del gráfico de precio
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos el generador de números aleatorios
   MathSrand(GetTickCount());
//--- generamos una muestra de la magnitud aleatoria
   long chart=0;
   string name="GraphicNormal";
   int n=100000;        // número de valores en la muestra
   int ncells=13;        // número de intervalos en el histograma
   double x[];           // centros de los intervalos del histograma
   double y[];           // número de valores de la muestra que han entrado en el inter
   double data[];        // muestra de valores aleatorios
   double max,min;       // valor máximo y mínimo en la muestra
//--- obtenemos la muestra de la distribución de Poisson
   MathRandomPoisson(lambda_par,n,data);
//--- calculamos los datos para construir el histograma
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtenemos los límites de la secuencia y el salto para construir la curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- obtenemos los datos calculados teóricamente en el intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(0,int(MathCeil(max)),1,x2);
   MathProbabilityDensityPoisson(x2,lambda_par,false,y2);
//--- escalamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- mostramos el gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Poisson distribution lambda=%G",lambda_par));
   graphic.BackgroundMainSize(16);
//--- desactivamos el escalado automático del eje Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,2));
   graphic.YAxis().Min(0);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- y ahora construimos la curva teórica de la densidad de la distribución
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- establecemos los centros de los intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- rellenamos las frecuencias de entrada en el intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos la amplitud absoluta de la secuencia, para obtener la precisión de no
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos los valores máximos y mínimos con la precisión establecida
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- el salto de generación de la secuencia también lo estableceremos a partir de la
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityPoisson

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución de Poisson con el parámetro lambda para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityPoisson(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  lambda,         // parámetro de distribución (mean)
   const bool    log_mode,       // cálculo del logaritmo del valor, si log_mode=true,
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución de Poisson con el parámetro lambda para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathProbabilityDensityPoisson(
   const double  x,              // valor de la magnitud aleatoria (número entero)
   const double  lambda,         // parámetro de distribución (mean)
   int&          error_code      // variable para el código de error
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución de Poisson con el parámetro lambda para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dhyper() en R.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  lambda,         // parámetro de distribución (mean)
   const bool    log_mode,       // bandera para calcular el logaritmo del valor, si
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Calcula el valor de la función de masa de probabilidad (probability mass function) de la distribución de Poisson con el parámetro lambda para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // matriz con los valores de la magnitud aleatoria
   const double  lambda,         // parámetro de distribución (mean)
   double&       result[]        // matriz para los valores de la función de densidad
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

lambda

[in] Parámetro de distribución (mean).

© 2000-2025, MetaQuotes Ltd.

log_mode

[in] Bandera para calcular el logaritmo del valor. Si log_mode=true, se retorna el logaritmo natural de densidad de la probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de densidad de probabilidad

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionPoisson

Calcula el valor de la función de distribución de Poisson con el parámetro lambda para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionPoisson(
   const double  x,             // valor de la magnitud aleatoria (número entero)
   const double  lambda,        // parámetro de distribución (mean)
   const double  tail,          // bandera para calcular, si lower_tail=true, entonces
   const bool    log_mode,      // bandera para calcular el logaritmo del valor, si lo
   int&          error_code     // variable para el código de error
   );
```

Calcula el valor de la función de distribución de Poisson con el parámetro lambda para una magnitud aleatoria x. En caso de error, retorna NaN.

```
double  MathCumulativeDistributionPoisson(
   const double  x,             // valor de la magnitud aleatoria (número entero)
   const double  lambda,        // parámetro de distribución (mean)
   int&          error_code     // variable para el código de error
   );
```

Calcula el valor de la función de distribución de Poisson con el parámetro lambda para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false. Análogo de dhyper() en R.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],           // matriz con los valores de la magnitud aleatoria
   const double  lambda,        // parámetro de distribución (mean)
   const double  tail,          // bandera para calcular, si lower_tail=true, entonce
   const bool    log_mode,      // bandera para calcular el logaritmo del valor, si l
   double&       result[]       // matriz para los valores de la función de distribuc
   );
```

Calcula el valor de la función de distribución de Poisson con el parámetro lambda para una matriz de magnitudes aleatorias x[]. En caso de error, retorna false.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],           // matriz con los valores de la magnitud aleatoria
   const double  lambda,        // parámetro de distribución (mean)
   double&       result[]       // matriz para los valores de la función de distribuc
   );
```

Parámetros

x

[in] Valor de la magnitud aleatoria.

x[]

[in] Matriz con los valores de la magnitud aleatoria.

lambda

[in] Parámetro de distribución (mean).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Bandera para calcular, si lower_tail=true, entonces se calcula la probabilidad de que la magnitud aleatoria no supere x.

log_mode

[in] Bandera para calcular el logaritmo del valor, si log_mode=true, entonces se calcula el logaritmo natural de probabilidad.

error_code

[out] Variable para anotar el código de error.

result[]

[out] Matriz para los valores de la función de distribución.

© 2000-2025, MetaQuotes Ltd.

MathQuantilePoisson

Calcula el valor inverso de la función de distribución de Poisson con el parámetro lambda para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantilePoisson(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  lambda,         // parámetro de distribución (mean)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor inverso de la función de distribución de Poisson con el parámetro lambda para la probabilidad *probability*. En caso de error, retorna NaN.

```
double  MathQuantilePoisson(
   const double  probability,    // valor de la probabilidad de aparición de una magni
   const double  lambda,         // parámetro de distribución (mean)
   int&          error_code      // variable para anotar el código de error
   );
```

Calcula el valor inverso de la función de distribución de Poisson con el parámetro lambda para una matriz de valores de probabilidad *probability*. En caso de error, retorna false. Análogo de qhyper() en R.

```
double  MathQuantilePoisson(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  lambda,         // parámetro de distribución (mean)
   const bool    tail,           // bandera para calcular, si lower_tail=false, entonc
   const bool    log_mode,       // bandera para calcular, si log_mode=true, entonces
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Calcula el valor inverso de la función de distribución de Poisson con el parámetro lambda para una matriz de valores de probabilidad *probability*. En caso de error, retorna false.

```
bool  MathQuantilePoisson(
   const double& probability[],  // matriz con los valores de probabilidad de la magni
   const double  lambda,         // parámetro de distribución (mean)
   double&       result[]        // matriz con los valores de los cuantiles
   );
```

Parámetros

probability

[in] Valor de la probabilidad de la magnitud aleatoria.

probability[]

[in] Matriz con los valores de probabilidad de la magnitud aleatoria.

lambda

© 2000-2025, MetaQuotes Ltd.

[in] Parámetro de distribución (mean).

tail

[in] bandera para calcular, si tail=false, entonces el cálculo se realiza para la probabilidad 1.0-probability.

log_mode

[in] Bandera para calcular, si log_mode=true, entonces el cálculo se realiza para la probabilidad Exp(probability).

error_code

[out] Variable para obtener el código de error.

result[]

[out] Matriz con los valores de los cuantiles.

© 2000-2025, MetaQuotes Ltd.

MathRandomPoisson

Genera una magnitud pseudoaleatoria, distribuida según la ley de la distribución de Poisson con el parámetro lambda. En caso de error, retorna NaN.

```
double  MathRandomPoisson(
   const double  lambda,          // parámetro de distribución (mean)
   int&          error_code       // variable para anotar el código de error
   );
```

Genera magnitudes pseudoaleatorias, distribuidas según la ley de la distribución de Poisson con el parámetro lambda. En caso de error, retorna false. Análogo de rgeom() en R.

```
bool  MathRandomPoisson(
   const double  lambda,          // parámetro de distribución (mean)
   const int     data_count,      // número de datos necesarios
   double&       result[]         // matriz con los valores de las magnitudes pseudoale
   );
```

Parámetros

lambda

[in] Parámetro de distribución (mean).

error_code

[out] Variable para anotar el código de error.

data_count

[out] Número de datos necesarios.

result[]

[out] Matriz para obtener los valores de las magnitudes pseudoaleatorias.

© 2000-2025, MetaQuotes Ltd.

MathMomentsPoisson

Calcula los valores numéricos teóricos de los 4 primeros momentos de la distribución de Poisson con el parámetro lambda.

```
double  MathMomentsPoisson(
   const double  lambda,          // parámetro de distribución (mean)
   double&       mean,            // variable para el valor medio
   double&       variance,        // variable para la varianza
   double&       skewness,        // variable para el coeficiente de asimetría
   double&       kurtosis,        // variable para la curtosis
   int&          error_code       // variable para el código de error
   );
```

Parámetros

lambda

[in] Parámetro de distribución (mean).

mean

[out] Variable para obtener el valor medio.

variance

[out] Variable para obtener la varianza.

skewness

[out] Variable para obtener el coeficiente de asimetría.

kurtosis

[out] Variable para obtener la curtosis.

error_code

[out] Variable para obtener el código de error.

Valor devuelto

Retorna true, si el cálculo de los momentos se realiza con éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

Funciones auxiliares

Grupo de funciones que realizan las operaciones matemáticas básicas: cálculo de la función gamma, de la función beta, el factorial, funciones exponenciales, logaritmos con diferentes bases, raíces cuadradas, etcétera.

Se presentan las posibilidades de procesamiento tanto de valores numéricos aparte (reales o enteros), como de sus matrices (con grabación de los resultados en una matriz aparte o en la matriz original).

Función	Descripción
MathRandomNonZero	Retorna un número aleatorio en el rango de 0.0 a 1.0 con coma flotante.
MathMoments	Calcula los primeros 4 momentos de la matriz: media, varianza, coeficiente de asimetría, curtosis.
MathPowInt	Eleva el número a la potencia entera indicada.
MathFactorial	Calcula el factorial del número entero indicado.
MathTrunc	Calcula la parte entera de una cifra dada o de los elementos de la matriz.
MathRound	Redondea un número o matriz numérica hasta el número dado de series fraccionadas.
MathArctan2	Calcula el ángulo cuya tangente es igual a la relación de los dos números indicados en el rango [-pi, pi].
MathGamma	Calcula el valor de la función gamma.
MathGammaLog	Calcula el logaritmo de la función gamma.
MathBeta	Calcula el valor de la función beta.
MathBetaLog	Calcula el valor del logaritmo de la función beta.
MathBetaIncomplete	Calcula el valor de la función beta incompleta.
MathGammaIncomplete	Calcula el valor de la función gamma incompleta.
MathBinomialCoefficient	Calcula el coeficiente binomial.
MathBinomialCoefficientLog	Calcula logaritmo del coeficiente binomial.
MathHypergeometric2F2	Calcula el valor de la función hipergeométrica.
MathSequence	Forma una secuencia basada en los valores: primer elemento, último elemento, salto de secuencia.

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathSequenceByCount	Forma una secuencia basada en los valores: primer elemento, último elemento, número de elementos de la secuencia.
MathReplicate	Forma una secuencia repetida de valores.
MathReverse	Forma una matriz de valores con un orden inverso de elementos.
MathIdentical	Compara dos matrices de valores y retorna true si coinciden todos los elementos.
MathUnique	Forma una matriz solo con los valores no repetidos.
MathQuickSortAscending	Función para la clasificación ascendente.
MathQuickSortDescending	Función para la clasificación descendente.
MathQuickSort	Función para la clasificación.
MathOrder	Forma una matriz con permutación de acuerdo con el orden de los elementos de la matriz después de la clasificación.
MathBitwiseNot	Calcula el resultado de la operación binaria NOT para los elementos de la matriz.
MathBitwiseAnd	Calcula el resultado de la operación binaria AND para los elementos de las matrices.
MathBitwiseOr	Calcula el resultado de la operación binaria OR para los elementos de las matrices.
MathBitwiseXor	Calcula el resultado de la operación binaria XOR para los elementos de las matrices.
MathBitwiseShiftL	Calcula el resultado de la operación binaria SHL para los elementos de la matriz.
MathBitwiseShiftR	Calcula el resultado de la operación binaria SHR para los elementos de la matriz.
MathCumulativeSum	Forma una matriz con la suma acumulada.
MathCumulativeProduct	Forma una matriz con el producto acumulado.
MathCumulativeMin	Forma una matriz con los valores mínimos acumulados.
MathCumulativeMax	Forma una matriz con los valores máximos acumulados.
MathSin	Calcula el valor de la función sin(x) para los elementos de la matriz.

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathCos	Calcula el valor de la función cos(x) para los elementos de la matriz.
MathTan	Calcula el valor de la función tan(x) para los elementos de la matriz.
MathArcsin	Calcula el valor de la función arcsin(x) para los elementos de la matriz.
MathArccos	Calcula el valor de la función arccos(x) para los elementos de la matriz.
MathArctan	Calcula el valor de la función arctan(x) para los elementos de la matriz.
MathSinPi	Calcula el valor de la función sin(pi*x) para los elementos de la matriz.
MathCosPi	Calcula el valor de la función cos(pi*x) para los elementos de la matriz.
MathTanPi	Calcula el valor de la función tan(pi*x) para los elementos de la matriz.
MathAbs	Calcula el valor absoluto de los elementos de la matriz.
MathCeil	Retorna el valor númerico entero superior más cercano para los elementos de la matriz.
MathFloor	Retorna el valor númerico entero inferior más cercano para los elementos de la matriz.
MathSqrt	Calcula la raíz cuadrada para los elementos de la matriz.
MathExp	Calcula el valor de la función exp(x) para los elementos de la matriz.
MathPow	Calcula el valor de la función pow(x, power) para los elementos de la matriz.
MathLog	Calcula el valor de la función log(x) para los elementos de la matriz.
MathLog2	Calcula el valor del logaritmo de base 2 para los elementos de la matriz.
MathLog10	Calcula el valor del logaritmo de base 10 para los elementos de la matriz.
MathDifference	Forma una matriz con las diferencias de los elementos y[i]=x[i+lag]-x[i].
MathSample	Realiza un muestreo aleatorio de los elementos de la matriz.

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathTukeySummary	Calcula el resumen de los 5 números de Tukey para los elementos de la matriz.
MathRange	Calcula los valores máximos y mínimos de los elementos de la matriz.
MathMin	Retorna el valor mínimo entre todos los elementos de la matriz.
MathMax	Retorna el valor máximo entre todos los elementos de la matriz.
MathSum	Retorna la suma de los elementos de la matriz.
MathProduct	Retorna el producto de los elementos de la matriz.
MathStandardDeviation	Calcula la desviación estándar de los elementos de la matriz.
MathAverageDeviation	Calcula la desviación media de los elementos de la matriz.
MathMedian	Calcula el valor mediano de los elementos de la matriz.
MathMean	Calcula el valor medio de los elementos de la matriz.
MathVariance	Calcula la varianza de los elementos de la matriz
MathSkewness	Calcula el coeficiente de asimetría de los elementos de la matriz.
MathKurtosis	Calcula la curtosis de los elementos de la matriz.
MathLog1p	Calcula el valor de la función log(1+x) para los elementos de la matriz.
MathExpm1	Calcula el valor de la función exp(x)-1 para los elementos de la matriz.
MathSinh	Calcula el valor de la función sinh(x) para los elementos de la matriz.
MathCosh	Calcula el valor de la función cosh(x) para los elementos de la matriz.
MathTanh	Calcula el valor de la función tanh(x) para los elementos de la matriz.
MathArcsinh	Calcula el valor de la función arcsinh(x) para los elementos de la matriz.

© 2000-2025, MetaQuotes Ltd.

Función	Descripción
MathArccosh	Calcula el valor de la función arccosh(x) para los elementos de la matriz.
MathArctanh	Calcula el valor de la función arctanh(x) para los elementos de la matriz.
MathSignif	Redondea el valor hasta el número indicado de signos en la mantisa.
MathRank	Calcula los rangos de los elementos de la matriz.
MathCorrelationPearson	Calcula el coeficiente de correlación de Pearson.
MathCorrelationSpearman	Calcula el coeficiente de correlación de Spearman.
MathCorrelationKendall	Calcula el coeficiente de correlación de Kendall.
MathQuantile	Calcula los cuantiles de muestreo correspondientes a las probabilidades indicadas.
MathProbabilityDensityEmpirical	Calcula la densidad empírica de la probabilidad para los valores aleatorios.
MathCumulativeDistributionEmpirical	Calcula la distribución empírica acumulativa para los valores aleatorios.

© 2000-2025, MetaQuotes Ltd.

MathRandomNonZero

Retorna un número aleatorio en el rango de 0.0 a 1.0 con coma flotante.

```
double  MathRandomNonZero()
```

Valor devuelto

Número aleatorio en el rango de 0.0 a 1.0 con coma flotante.

© 2000-2025, MetaQuotes Ltd.

MathMoments

Calcula los primeros 4 momentos de la matriz: media, varianza, coeficiente de asimetría, curtosis.

```
bool  MathMoments(
   const double&  array[],            // matriz de valores
   double&        mean,               // variable para la media
   double&        variance,           // variable para la varianza
   double&        skewness,           // variable para el coeficiente de asimetría
   double&        kurtosis,           // variable para la curtosis
   const int      start=0,            // índice inicial
   const int      count=WHOLE_ARRAY   // número de elementos
   )
```

Parámetros

array[]

[in] Matriz de valores.

mean

[out] Variable para el valor medio (1 momento).

variance

[out] Variable para la varianza (2 momento).

skewness

[out] Variable para el coeficiente de asimetría (3 momento).

kurtosis

[out] Variable para la curtosis (4 momento).

start=0

[in] Índice inicial para el cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para el cálculo.

Valor devuelto

Retorna true, si los momentos se han calculado con éxito, si no, false.

© 2000-2025, MetaQuotes Ltd.

MathPowInt

Eleva el número a la potencia entera indicada.

```
double  MathPowInt(
   const double  x,       // valor del número
   const int     power    // exponenciación
   )
```

Parámetros

x

[in] Número de precisión doble con coma flotante, elevada a la potencia.

power

[in] Número entero que establece la potencia.

Valor devuelto

Número x, elevado a la potencia indicada.

© 2000-2025, MetaQuotes Ltd.

MathFactorial

Calcula el factorial del número entero indicado.

```
double  MathFactorial(
   const int  n   // valor del número
   )
```

Parámetros

n

[in] Número entero cuyo factorial hay que calcular.

Valor devuelto

Factorial del número.

© 2000-2025, MetaQuotes Ltd.

MathTrunc

Calcula la parte entera de una cifra dada o de los elementos de la matriz.

Versión para trabajar con un número de precisión doble con coma flotante:

```
double  MathTrunc(
   const double   x          // valor del número
   )
```

Valor devuelto

Parte entera del número indicado.

Versión para trabajar con una matriz de números de precisión doble con coma flotante: Los resultados se graban en una nueva matriz:

```
bool  MathTrunc(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Versión para trabajar con una matriz de números de precisión doble con coma flotante: Los resultados se graban en esta misma matriz:

```
bool  MathTrunc(
   double&        array[]    // matriz de valores
   )
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Parámetros

x

[in] Número de precisión doble con coma flotante, cuya parte entera hay que obtener.

array[]

[in] Matriz de números de precisión doble con coma flotante, cuya parte entera hay que obtener.

array[]

[out] Matriz de valores de salida.

result[]

[out] Matriz de valores de salida.

© 2000-2025, MetaQuotes Ltd.

MathRound

Redondea un número de precisión doble con coma flotante o una matriz de tales números hasta el número dado de series fraccionadas.

Versión para redondear los números de precisión doble con coma flotante hasta el número dado de series fraccionadas:

```
double  MathRound(
   const double   x,          // valor del número
   const int      digits      // número de decimales tras la coma
   )
```

Valor devuelto

Un número cercano al parámetro x, cuya cantidad de cifras de la parte fraccionada es igual a digits.

Versión para redondear matrices de números de precisión doble con coma flotante hasta el número dado de series fraccionadas: Los resultados se graban en una nueva matriz.

```
bool  MathRound(
   const double&  array[],    // matriz de valores
   int            digits,     // número de dígitos tras la coma
   double&        result[]    // matriz de resultados
   )
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Versión para redondear matrices de números de precisión doble con coma flotante hasta el número dado de series fraccionadas: Los resultados se graban en esta misma matriz.

```
bool  MathRound(
   double&        array[],    // matriz de valores
   int            digits      // número de decimales tras la coma
   )
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Parámetros

x

[in] Número de precisión doble a redondear con coma flotante.

digits

[in] Número de dígitos de la parte fraccionada en el valor retornado.

array[]

[in] Matriz de los números de precisión doble a redondear con coma flotante.

© 2000-2025, MetaQuotes Ltd.

array[]

[out] Matriz de los valores de salida.

result[]

[out] Matriz de los valores de salida.

© 2000-2025, MetaQuotes Ltd.

MathArctan2

Retorna el arcotangente del cociente de dos argumentos (x, y).

Versión para trabajar con la relación de dos números indicados (x, y):

```
double  MathArctan2(
   const double       y,          // coordenada y
   const double       x           // coordenada x
   )
```

Valor devuelto

Ángulo, θ, medido en radianes, de tal forma que -π≤θ≤π, y tan (θ) = y o x, donde (x, y) es el punto en el sistema de coordenadas cartesianas.

Versión para trabajar con la relación de una pareja de elementos de las matrices x e y:

```
bool  MathArctan2(
   const double&      x[],        // matriz de valores x
   const double&      y[],        // matriz de valores y
   double&            result[]    // matriz de resultados
   )
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Parámetros

y

[in] Coordenada y del punto.

x

[in] Coordenada x del punto.

x[]

[in] Matriz de las coordenadas x de los puntos.

y[]

[in] Matriz de las coordenadas y de los puntos.

result[]

[out] Matriz para la grabación de resultados

Observaciones

Preste atención a lo siguiente.

- Para (x, y) en el cuadrante 1 el valor retornado será: 0 < θ < π/2.
- Para (x, y) en el cuadrante 2 el valor retornado será: π/2 < θ≤π.
- Para (x, y) en el cuadrante 3 el valor retornado será: -π < θ <-π/2.
- Para (x, y) en el cuadrante 4 el valor retornado será: -π/2 < θ < 0.

© 2000-2025, MetaQuotes Ltd.

Para los puntos fuera de los cuadrantes indicados, el valor retornado se indica más abajo.

- Si y es igual a 0 y x no es negativa, θ = 0.
- Si y es igual a 0 y x es negativa, θ = π.
- Si y es un número positivo, y x es igual a 0, θ = π/2.
- Si y es un número negativo y x es igual a 0, θ = -π/2.
- Si y es igual a 0 y x es igual 0, entonces θ = -π/2.

Si el valor del parámetro x o y es igual a NaN o bien si los valores de los parámetros x e y son iguales al valor de PositiveInfinity o NegativeInfinity, el método retorna el valor NaN.

© 2000-2025, MetaQuotes Ltd.

MathGamma

Calcula el valor de la función gamma para el argumento real *x*.

```
double  MathGamma(
   const double  x       // argumento de la función
   )
```

Parámetros

x

[in] Argumento real de la función.

Valor devuelto

Valor de la función gamma.

© 2000-2025, MetaQuotes Ltd.

MathGammaLog

Calcula el valor del logaritmo de la función gamma para el argumento real x.

```
double  MathGammaLog(
   const double  x      // argumento de la función
   )
```

Parámetros

x

[in] Argumento real de la función.

Valor devuelto

Valor del logaritmo de la función.

© 2000-2025, MetaQuotes Ltd.

MathBeta

Calcula el valor de la función beta para los argumentos reales a y b.

```
double  MathBeta(
   const double  a,      // primer argumento de la función
   const double  b       // segundo argumento de la función
   )
```

Parámetros

a

[in] Argumento de la función a.

b

[in] Argumento de la función b.

Valor devuelto

Valor de la función.

© 2000-2025, MetaQuotes Ltd.

MathBetaLog

Calcula el valor del logaritmo de la función beta para los argumentos reales a y b.

```
double  MathBetaLog(
   const double  a,      // primer argumento de la función
   const double  b       // segundo argumento de la función
   )
```

Parámetros

a

[in] Argumento de la función a.

b

[in] Argumento de la función b.

Valor devuelto

Valor del logaritmo de la función.

© 2000-2025, MetaQuotes Ltd.

MathBetaIncomplete

Calcula el valor de la función beta incompleta.

```
double  MathBetaIncomplete(
   const double  x,      // argumento de la función
   const double  p,      // primer parámetro de la función
   const double  q       // segundo parámetro de la función
   )
```

Parámetros

x

[in] Argumento de la función.

p

[in] El primer parámetro de la función beta debe ser >0.0.

q

[in] El segundo parámetro de la función beta debe ser >0.0.

Valor devuelto

Valor de la función.

© 2000-2025, MetaQuotes Ltd.

MathGammaIncomplete

Calcula el valor de la función gamma incompleta.

```
double  MathGammaIncomplete(
   double  x,          // argumento de la función
   double  alpha       // parámetro de la función
   )
```

Parámetros

x

[in] Argumento de la función.

alpha

[in] Parámetro de la función gamma incompleta.

Valor devuelto

Valor de la función.

© 2000-2025, MetaQuotes Ltd.

MathBinomialCoefficient

Calcula el coeficiente binomial: C(n,k)=n!/(k!*(n-k)!).

```
long  MathBinomialCoefficient(
   const int  n,       // número total de elementos
   const int  k        // número de elementos combinados
   )
```

Parámetros

n

[in] Número de elementos.

k

[in] Número de elementos para cada combinación.

Valor devuelto

Número de combinaciones a partir de N según K.

© 2000-2025, MetaQuotes Ltd.

MathBinomialCoefficientLog

Calcula el logaritmo del coeficiente binomial: Log(C(n,k))=Log(n!/(k!*(n-k)!))

Versión para los argumentos enteros:

```
double  MathBinomialCoefficientLog(
   const int      n,       // número total de elementos
   const int      k        // número de elementos combinados
   )
```

Valor para los argumentos reales:

```
double  MathBinomialCoefficientLog(
   const double   n,       // número total de elementos
   const double   k        // número de elementos combinados
   )
```

Parámetros

n

[in] Número de elementos.

k

[in] Número de elementos para cada combinación.

Valor devuelto

Logaritmo de C(n,k).

© 2000-2025, MetaQuotes Ltd.

MathHypergeometric2F2

Calcula el valor de la función Hypergeometric_2F2 (a, b, c, d, z), usando el método de Taylor.

```
double  MathHypergeometric2F2(
   const double  a,      // primer parámetro de la función
   const double  b,      // segundo parámetro de la función
   const double  c,      // tercer parámetro de la función
   const double  d,      // cuarto parámetro de la función
   const double  z       // quinto parámetro de la función
   )
```

Parámetros

a

[in] Primer parámetro de la función.

b

[in] Segundo parámetro de la función.

c

[in] Tercer parámetro de la función.

d

[in] Cuarto parámetro de la función.

z

[in] Quinto parámetro de la función.

Valor devuelto

Valor de la función.

© 2000-2025, MetaQuotes Ltd.

MathSequence

Forma una secuencia de valores basada en los datos de los valores: primer elemento, último elemento, salto de secuencia.

Versión para trabajar con valores reales:

```
bool  MathSequence(
   const double  from,        // valor inicial
   const double  to,          // valor final
   const double  step,        // salto
   double&       result[]     // matriz de resultados
   )
```

Versión para trabajar con valores enteros:

```
bool  MathSequence(
   const int     from,        // valor inicial
   const int     to,          // valor final
   const int     step,        // salto
   int&          result[]     // matriz de resultados
   )
```

Parámetros

from

[in] Primer valor de la secuencia

to

[in] Último valor de la secuencia

step

[in] Salto de la secuencia.

result[]

[out] Matriz para grabar la secuencia.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathSequenceByCount

Forma una secuencia de valores basada en los datos de los valores: primer elemento, último elemento, número de elementos de la secuencia.

Versión para trabajar con valores reales:

```
bool  MathSequenceByCount(
   const double  from,        // valor inicial
   const double  to,          // valor final
   const int     count,       // número
   double&       result[]     // matriz de resultados
   )
```

Versión para trabajar con valores enteros:

```
bool  MathSequenceByCount(
   const int     from,        // valor inicial
   const int     to,          // valor final
   const int     count,       // número
   int&          result[]     // matriz de resultados
   )
```

Parámetros

from

[in] Primer valor de la secuencia.

to

[in] Último valor de la secuencia.

count

[in] Número de elementos de la secuencia.

result[]

[out] Matriz para grabar la secuencia.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathReplicate

Forma una secuencia repetida de valores.

Versión para trabajar con valores reales:

```
bool  MathReplicate(
   const double&  array[],    // matriz de valores
   const int      count,      // número de repeticiones
   double&        result[]    // matriz de resultados
   )
```

Versión para trabajar con valores enteros:

```
bool  MathReplicate(
   const int&     array[],    // matriz de valores
   const int      count,      // número de repeticiones
   int&           result[]    // matriz de resultados
   )
```

Parámetros

array[]

[in] Matriz para generar la secuencia.

count

[in] Número de repeticiones de la matriz en la secuencia.

result[]

[out] Matriz para grabar la secuencia.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathReverse

Forma una matriz de valores con un orden inverso de elementos.

Versión para trabajar con valores reales con guardado de los resultados en una matriz nueva:

```
bool  MathReverse(
   const double&  array[],      // matriz de valores
   double&        result[]      // matriz de resultados
   )
```

Versión para trabajar con valores enteros con guardado de los resultados en una matriz nueva:

```
bool  MathReverse(
   const int&     array[],      // matriz de valores
   int&           result[]      // matriz de resultados
   )
```

Versión para trabajar con valores reales con guardado de los resultados en la misma matriz:

```
bool  MathReverse(
   double&        array[]       // matriz de valores
   )
```

Versión para trabajar con valores enteros con guardado de los resultados en la misma matriz:

```
bool  MathReverse(
   int&           array[]       // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

array[]

[out] Matriz de salida con orden de valores inverso.

result[]

[out] Matriz de salida con orden de valores inverso.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathIdentical

Compara dos matrices de valores y retorna true si coinciden todos los elementos.

Versión para trabajar con matrices de valores reales:

```
bool  MathIdentical(
   const double&  array1[],      // primera matriz de valores
   const double&  array2[]       // segunda matriz de valores
   )
```

Versión para trabajar con matrices de valores enteros:

```
bool  MathIdentical(
   const int&     array1[],      // primera matriz de valores
   const int&     array2[]       // segunda matriz de valores
   )
```

Parámetros

array1[]

[in] Primera matriz para la comparación.

array2[]

[in] Segunda matriz para la comparación.

Valor devuelto

Retorna true si las matrices son iguales, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

MathUnique

Forma una matriz solo con los valores no repetidos.

Versión para trabajar con valores reales:

```
bool  MathUnique(
   const double&   array[],   // matriz de valores
   double&         result[]   // matriz de resultados
   )
```

Versión para trabajar con valores enteros:

```
bool  MathUnique(
   const int&     array[],      // matriz de valores
   int&           result[]      // matriz de resultados
   )
```

Parámetros

array[]

[in] Matriz original.

result[]

[out] Matriz para grabar valores únicos.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathQuickSortAscending

Matriz para clasificar de forma ascendente las matrices array[] y indices[], usando el algoritmo QuickSort.

```
void  MathQuickSortAscending(
   double&  array[],     // matriz de valores
   int&     indices[],   // matriz de índices
   int      first,       // valor inicial
   int      last         // valor final
   )
```

Parámetros

array[]

[in][out] Matriz para la clasificación.

indices[]

[in][out] Matriz para guardar los índices de la matriz original.

first

[in] Índice del elemento desde el que se debe comenzar la clasificación.

last

[in] Índice del elemento en el que hay que concluir la clasificación.

© 2000-2025, MetaQuotes Ltd.

MathQuickSortDescending

Matriz para clasificar de forma descendente las matrices array[] y indices[], usando el algoritmo QuickSort.

```
void  MathQuickSortDescending(
   double&  array[],     // matriz de valores
   int&     indices[],   // matriz de índices
   int      first,       // valor inicial
   int      last         // valor final
   )
```

Parámetros

array[]

[in][out] Matriz para la clasificación.

indices[]

[in][out] Matriz para guardar los índices de la matriz original.

first

[in] Índice del elemento desde el que se debe comenzar la clasificación.

last

[in] Índice del elemento en el que hay que concluir la clasificación.

© 2000-2025, MetaQuotes Ltd.

MathQuickSort

Matriz para clasificar simultáneamente las matrices array[] e indices[], usando el algoritmo QuickSort.

```
void  MathQuickSort(
   double&  array[],      // matriz de valores
   int&     indices[],    // matriz de índices
   int      first,        // valor inicial
   int      last,         // valor final
   int      mode          // dirección
   )
```

Parámetros

array[]

[in][out] Matriz para la clasificación.

indices[]

[in][out] Matriz para guardar los índices de la matriz original.

first

[in] Índice del elemento desde el que se debe comenzar la clasificación.

last

[in] Índice del elemento en el que hay que concluir la clasificación.

mode

[in] Dirección de la clasificación (>0 ascendente, de lo contrario, descendente).

© 2000-2025, MetaQuotes Ltd.

MathOrder

Forma una matriz de número entero con permutación de acuerdo con el orden de los elementos de la matriz después de la clasificación.

Versión para trabajar con la matriz de valores reales:

```
bool  MathOrder(
   const double&  array[],   // matriz de valores
   int&           result[]   // matriz de resultados
   )
```

Versión para trabajar con la matriz de valores enteros:

```
bool  MathOrder(
   const int&     array[],   // matriz de valores
   int&           result[]   // matriz de resultados
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz para grabar los índices clasificados.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseNot

Calcula el resultado de la operación binaria NOT para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathBitwiseNot(
   const int&  array[],   // matriz de valores
   int&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathBitwiseNot(
   int&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

array[]

[out] Matriz de valores de salida.

result[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseAnd

Calcula el resultado de la operación binaria AND para las matrices indicadas.

```
bool  MathBitwiseAnd(
   const int&  array1[],  // primera matriz de valores
   const int&  array2[],  // segunda matriz de valores
   int&        result[]   // matriz de resultados
   )
```

Parámetros

array1[]

[in] Primera matriz de valores.

array2[]

[in] Segunda matriz de valores.

result[]

[out] Matriz para la grabación de resultados.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseOr

Calcula el resultado de la operación binaria OR para las matrices indicadas.

```
bool  MathBitwiseOr(
   const int&  array1[],  // primera matriz de valores
   const int&  array2[],  // segunda matriz de valores
   int&        result[]   // matriz de resultados
   )
```

Parámetros

array1[]

[in] Primera matriz de valores.

array2[]

[in] Segunda matriz de valores.

result[]

[out] Matriz para la grabación de resultados.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseXor

Calcula el resultado de la operación binaria XOR para las matrices indicadas.

```
bool  MathBitwiseXor(
   const int&  array1[],  // primera matriz de valores
   const int&  array2[],  // segunda matriz de valores
   int&        result[]   // matriz de resultados
   )
```

Parámetros

array1[]

[in] Primera matriz de valores.

array2[]

[in] Segunda matriz de valores.

result[]

[out] Matriz para la grabación de resultados.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseShiftL

Calcula el resultado de la operación binaria SHL (desplazamiento en bits a la izquierda) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathBitwiseShiftL(
   const int&  array[],     // matriz de valores
   const int   n,           // valor del desplazamiento
   int&        result[]     // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathBitwiseShiftL(
   int&        array[],     // matriz de valores
   const int   n            // valor del desplazamiento
   )
```

Parámetros

array[]

[in] Matriz de valores.

n

[in] Número de bits para el desplazamiento.

array[]

[out] Matriz de valores de salida.

result[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseShiftR

Calcula el resultado de la operación binaria SHR (desplazamiento en bits a la derecha) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathBitwiseShiftR(
   const int&  array[],     // matriz de valores
   const int   n,           // valor del desplazamiento
   int&        result[]     // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathBitwiseShiftR(
   int&        array[],     // matriz de valores
   const int   n            // valor del desplazamiento
   )
```

Parámetros

array[]

[in] Matriz de valores.

n

[in] Número de bits para el desplazamiento.

array[]

[out] Matriz de valores de salida.

result[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeSum

Forma una matriz con la suma acumulada.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathCumulativeSum(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathCumulativeSum(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

array[]

[out] Matriz de valores de salida.

result[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeProduct

Forma una matriz con el producto acumulado.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathCumulativeProduct(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathCumulativeProduct(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeMin

Forma una matriz con los valores mínimos acumulados.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathCumulativeMin(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathCumulativeMin(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeMax

Forma una matriz con los valores máximos acumulados.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathCumulativeMax(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathCumulativeMax(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathSin

Calcula el valor de la función sin(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathSin(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathSin(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCos

Calcula el valor de la función cos(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathCos(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathCos(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathTan

Calcula el valor de la función tan(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathTan(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathTan(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathArcsin

Calcula el valor de la función arcsin(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathArcsin(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathArcsin(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathArccos

Calcula el valor de la función arccos(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathArccos(
   const double&   array[],    // matriz de valores
   double&         result[]    // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathArccos(
   double&         array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathArctan

Calcula el valor de la función arctan(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathArctan(
   const double&   array[],   // matriz de valores
   double&         result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathArctan(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathSinPi

Calcula el valor de la función sin(pi*x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathSinPi(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathSinPi(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCosPi

Calcula el valor de la función cos(pi*x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathCosPi(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathCosPi(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathTanPi

Calcula el valor de la función tan(pi*x) para los elementos de la matriz.

Versión con grabación del resultado en una nueva matriz:

```
bool  MathTanPi(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación del resultado en la matriz original:

```
bool  MathTanPi(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathAbs

Calcula el valor absoluto de los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathAbs(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathAbs(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCeil

Retorna el valor númerico entero superior más cercano para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathCeil(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathCeil(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathFloor

Retorna el valor númerico entero inferior más cercano para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathFloor(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathFloor(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathSqrt

Calcula la raíz cuadrada para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathSqrt(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathSqrt(
   double&         array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathExp

Calcula el valor de la función exp(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathExp(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathExp(
   double&       array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathPow

Calcula el valor de la función pow(x, power) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathPow(
   const double&  array[],    // matriz de valores
   const double   power,      // potencia
   double&        result[]    // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathPow(
   double&        array[],    // matriz de valores
   const double   power       // potencia
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathLog

Calcula el valor de la función log(x) para los elementos de la matriz.

Versión para calcular el logaritmo natural con grabación de los resultados en una nueva matriz:

```
bool  MathLog(
   const double&  array[],      // matriz de valores
   double&        result[]      // matriz de resultados
   )
```

Versión para calcular el logaritmo natural con grabación de los resultados en la matriz original:

```
bool  MathLog(
   double&        array[]       // matriz de valores
   )
```

Versión para calcular el logaritmo según la base establecida, con grabación de los resultados en una nueva matriz:

```
bool  MathLog(
   const double&   array[],     // matriz de valores
   const double    base,        // base del logaritmo
   double&         result[]     // matriz de resultados
   )
```

Versión para calcular el logaritmo según la base establecida, con grabación de los resultados en la matriz original:

```
bool  MathLog(
   double&         array[],     // matriz de valores
   const double    base         // base del logaritmo
   )
```

Parámetros

array[]

[in] Matriz de valores.

base

[in] Base del logaritmo.

array[]

[out] Matriz de valores de salida.

result[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathLog2

Calcula el valor del logaritmo de base 2 para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathLog2(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathLog2(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathLog10

Calcula el valor del logaritmo de base 10 para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathLog10(
   const double&   array[],   // matriz de valores
   double&         result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathLog10(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathLog1p

Calcula el valor de la función log(1+x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathLog1p(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathLog1p(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathDifference

Forma una matriz con las diferencias de los elementos y[i]=x[i+lag]-x[i].

Matriz para la generación única de la matriz de valores reales:

```
bool  MathDifference(
   const double  &array[],     // matriz de valores
   const int     lag,          // retraso
   double        &result[]     // matriz de resultados
   )
```

Matriz para la generación única de la matriz de valores enteros:

```
bool  MathDifference(
   const int     &array[],     // matriz de valores
   const int     lag,          // retraso
   int           &result[]     // matriz de resultados
   )
```

Matriz para la generación múltiple de la matriz de valores reales (el número de iteraciones se establece en los parámetros de entrada):

```
bool  MathDifference(
   const double  &array[],     // matriz de valores
   const int     lag,          // retraso
   const int     differences,  // número de iteraciones
   double        &result[]     // matriz de resultados
   )
```

Matriz para la generación múltiple de la matriz de valores enteros (el número de iteraciones se establece en los parámetros de entrada):

```
bool  MathDifference(
   const int&    array[],      // matriz de valores
   const int     lag,          // retraso
   const int     differences,  // número de iteraciones
   int&          result[]      // matriz de resultados
   )
```

Parámetros

array[]

[in] Matriz de valores.

lag

[in] Parámetro de retraso.

differences

[in] Número de iteraciones.

result[]

© 2000-2025, MetaQuotes Ltd.

[out] Matriz para la grabación de resultados.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathSample

Realiza un muestreo aleatorio de los elementos de la matriz.

Versión para trabajar con la matriz de valores reales:

```
bool  MathSample(
   const double&  array[],          // matriz de valores
   const int      count,            // número
   double&        result[]          // matriz de resultados
   )
```

Versión para trabajar con la matriz de valores enteros:

```
bool  MathSample(
   const int&     array[],          // matriz de valores
   const int      count,            // número
   int&           result[]          // matriz de resultados
   )
```

Versión para trabajar con la matriz de valores reales. Es posible obtener una muestra con retorno:

```
bool  MathSample(
   const double&  array[],          // matriz de valores
   const int      count,            // número
   const bool     replace,          // bandera
   double&        result[],         // matriz de resultados
   )
```

Versión para trabajar con la matriz de valores enteros. Es posible obtener una muestra con retorno:

```
bool  MathSample(
   const int&     array[],          // matriz de valores
   const int      count,            // número
   const bool     replace,          // bandera
   int&           result[]          // matriz de resultados
   )
```

Versión para trabajar con una matriz de valores reales, para los que se ha definido la probabilidad de entrada en la muestra.

```
bool  MathSample(
   const double&  array[],          // matriz de valores
   double&        probabilities[],  // matriz de probabilidades
   const int      count,            // número
   double&        result[]          // matriz de resultados
   )
```

Versión para trabajar con una matriz de valores enteros, para los que se ha definido la probabilidad de entrada en la muestra.

© 2000-2025, MetaQuotes Ltd.

```
bool  MathSample(
   const int&     array[],           // matriz de valores
   double&        probabilities[],   // matriz de probabilidades
   const int      count,             // número
   int&           result[]           // matriz de resultados
   )
```

Versión para trabajar con una matriz de valores reales, para los que se ha definido la probabilidad de entrada en la muestra. Es posible obtener una muestra con retorno:

```
bool  MathSample(
   const double&  array[],           // matriz de valores
   double&        probabilities[],   // matriz de probabilidades
   const int      count,             // número
   const bool     replace,           // bandera
   double&        result[]           // matriz de resultados
   )
```

Versión para trabajar con una matriz de valores enteros, para los que se ha definido la probabilidad de entrada en la muestra. Es posible obtener una muestra con retorno:

```
bool  MathSample(
   const int&     array[],           // matriz de valores
   double&        probabilities[],   // matriz de probabilidades
   const int      count,             // número
   const bool     replace,           // bandera
   int&           result[]           // matriz de resultados
   )
```

Parámetros

array[]

[in] matriz de valores enteros.

probabilities[]

[in] Matriz de probailidades con las que se realiza la muestra de elementos.

count

[in] Número de elementos.

replace

[in] Parámetro que permite realizar la selección con retorno.

result[]

[out] Matriz para grabar los resultados.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Nota

© 2000-2025, MetaQuotes Ltd.

El argumento replace=true permite realizar la selección aleatoria de elementos con su posterior retorno al conjunto original.

© 2000-2025, MetaQuotes Ltd.

MathTukeySummary

Calcula el resumen de los 5 números de Tukey (mínimo, cuartil inferior, medio, cuartil superior, máximo) para los elementos de la matriz.

```
bool  MathTukeySummary(
   const double&  array[],       // matriz de valores
   const bool     removeNAN,     // bandera
   double&        minimum,       // valor mínimo
   double&        lower_hinge,   // cuartil inferior
   double&        median,        // valor medio
   double&        upper_hinge,   // cuartil superior
   double&        maximum        // valor máximo
   )
```

Parámetros

array[]

[in] Matriz de valores reales.

removeNAN

[in] bandera que indica si hay que eliminar los valores no numéricos.

minimum

[out] Variable para grabar el valor mínimo.

lower_hinge

[out] Variable para grabar el cuartil inferior.

median

[out] Variable para grabar el valor medio.

upper_hinge

[out] Variable para grabar el cuartil superior.

maximum

[out] Variable para grabar el valor máximo.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathRange

Calcula los valores máximos y mínimos de los elementos de la matriz.

```
bool  MathRange(
   const double&  array[],  // matriz de valores
   double&        min,      // valor mínimo
   double&        max       // valor máximo
   )
```

Parámetros

array[]

[in] Matriz de valores.

min

[out] Variable para grabar el valor mínimo.

max

[out] Variable para grabar el valor máximo.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathMin

Retorna el valor mínimo entre todos los elementos de la matriz.

```
double  MathMin(
   const double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Valor mínimo.

© 2000-2025, MetaQuotes Ltd.

MathMax

Retorna el valor máximo entre todos los elementos de la matriz.

```
double  MathMax(
   const double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Valor máximo.

© 2000-2025, MetaQuotes Ltd.

MathSum

Retorna la suma de los elementos de la matriz.

```
double  MathSum(
   const double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Suma de los elementos.

© 2000-2025, MetaQuotes Ltd.

MathProduct

Retorna el producto de los elementos de la matriz.

```
double  MathProduct(
   const double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Producto de los elementos.

© 2000-2025, MetaQuotes Ltd.

MathStandardDeviation

Calcula la desviación estándar de los elementos de la matriz.

```
double  MathStandardDeviation(
   const double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Desviación estándar.

© 2000-2025, MetaQuotes Ltd.

MathAverageDeviation

La función calcula la desviación media de los elementos de la matriz.

```
double  MathAverageDeviation(
   const double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Desviación media de los elementos de la matriz.

© 2000-2025, MetaQuotes Ltd.

MathMedian

Calcula el valor mediano de los elementos de la matriz.

```
double  MathMedian(
   double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Valor mediano.

© 2000-2025, MetaQuotes Ltd.

MathMean

Calcula el valor medio de los elementos de la matriz.

```
double  MathMean(
   const double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Valor medio.

© 2000-2025, MetaQuotes Ltd.

MathVariance

La función calcula la varianza (segundo momento) de los elementos de la matriz.

```
double  MathVariance(
   const double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Valor de la varianza.

© 2000-2025, MetaQuotes Ltd.

MathSkewness

La función calcula el coeficiente de asimetría (tercer momento) de los elementos de la matriz.

```
double  MathSkewness(
   const double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Coeficiente de asimetría.

© 2000-2025, MetaQuotes Ltd.

MathKurtosis

La función calcula la curtosis (cuarto momento) de los elementos de la matriz.

```
double  MathKurtosis(
   const double&  array[]   // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

Valor devuelto

Curtosis.

© 2000-2025, MetaQuotes Ltd.

MathExpm1

Calcula el valor de la función exp(x)-1 para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathExpm1(
   const double&   array[],   // matriz de valores
   double&         result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathExpm1(
   double&         array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathSinh

Calcula el valor de la función sinh(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathSinh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathSinh(
   double&       array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCosh

Calcula el valor de la función cosh(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathCosh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathCosh(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathTanh

Calcula el valor de la función tanh(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathTanh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathTanh(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathArcsinh

Calcula el valor de la función arcsinh(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathArcsinh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathArcsinh(
   double&         array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathArccosh

Calcula el valor de la función arccosh(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathArccosh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathArccosh(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathArctanh

Calcula el valor de la función arctanh(x) para los elementos de la matriz.

Versión con grabación de los resultados en una nueva matriz:

```
bool  MathArctanh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultados
   )
```

Versión con grabación de los resultados en la matriz original:

```
bool  MathArctanh(
   double&        array[]    // matriz de valores
   )
```

Parámetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de salida.

array[]

[out] Matriz de valores de salida.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathSignif

Redondea el valor hasta el número indicado de signos en la mantisa.

Versión para trabajar con valores reales:

```
double  MathSignif(
   const double   x,          // valor
   const int      digits      // número de dígitos
   )
```

Valor devuelto

Valor redondeado.

Versión para trabajar con una matriz de valores reales con grabado de resultados en una matriz aparte:

```
bool  MathSignif(
   const double&  array[],    // matriz de valores
   int            digits,     // número de dígitos
   double         result[]    // matriz de resultados
   )
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Versión para trabajar con una matriz de valores reales con grabado de resultados en la matriz original:

```
bool  MathSignif(
   double&        array[],    // matriz de valores
   int            digits      // número знаков
   )
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

Parámetros

x

[in] Valor real para el redondeo.

digits

[in] Número de dígitos.

array[]

[in] Matriz de valores reales.

array[]

[out] Matriz de valores de salida.

result[]

[out] Matriz de valores de salida.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

MathRank

Calcula los rangos de los elementos de la matriz.

Versión para trabajar con la matriz de valores reales:

```
bool  MathRank(
   const double&  array[],  // matriz de valores
   double&        rank[]    // matriz de rangos
   )
```

Versión para trabajar con la matriz de valores enteros:

```
bool  MathRank(
   const int&     array[],  // matriz de valores
   double&        rank[]    // matriz de rangos
   )
```

Parámetros

array[]

[in] Matriz de valores.

rank[]

[out] Matriz para grabar los rangos.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCorrelationPearson

Calcula el coeficiente de correlación de Pearson.

Versión para trabajar con matrices de valores reales:

```
bool  MathCorrelationPearson(
   const double&  array1[],  // primera matriz de valores
   const double&  array2[],  // segunda matriz de valores
   double&        r          // coeficiente de correlación
   )
```

Versión para trabajar con matrices de valores enteros:

```
bool  MathCorrelationPearson(
   const int&     array1[],  // primera matriz de valores
   const int&     array2[],  // segunda matriz de valores
   double&        r          // coeficiente de correlación
   )
```

Parámetros

array1[]

[in] Primera matriz de valores.

array2[]

[in] Segunda matriz de valores.

r

[out] Variable para grabar el coeficiente de correlación.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCorrelationSpearman

Calcula el coeficiente de correlación de Spearman.

Versión para trabajar con matrices de valores reales:

```
bool  MathCorrelationSpearman(
   const double&  array1[],  // primera matriz de valores
   const double&  array2[],  // segunda matriz de valores
   double&        r          // coeficiente de correlación
   )
```

Versión para trabajar con matrices de valores enteros:

```
bool  MathCorrelationSpearman(
   const int&     array1[],   // primera matriz de valores
   const int&     array2[],   // segunda matriz de valores
   double&        r           // coeficiente de correlación
   )
```

Parámetros

array1[]

[in] Primera matriz de valores.

array2[]

[in] Segunda matriz de valores.

r

[out] Variable para grabar el coeficiente de correlación.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCorrelationKendall

Calcula el coeficiente de correlación de Kendall.

Versión para trabajar con matrices de valores reales:

```
bool  MathCorrelationKendall(
   const double&  array1[],  // primera matriz de valores
   const double&  array2[],  // segunda matriz de valores
   double&        tau        // Coeficiente de correlación
   )
```

Versión para trabajar con matrices de valores enteros:

```
bool  MathCorrelationKendall(
   const int&     array1[],  // primera matriz de valores
   const int&     array2[],  // segunda matriz de valores
   double&        tau        // coeficiente de correlación
   )
```

Parámetros

array1[]

[in] Primera matriz de valores.

array2[]

[in] Segunda matriz de valores.

tau

[out] Variable para grabar el coeficiente de correlación.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathQuantile

Calcula los cuantiles de muestreo, correspondientes a las probabilidades indicadas: Q[i](p) = (1 - gamma)*x[j] + gamma*x[j+1]

```
bool  MathQuantile(
   const double&  array[],     // matriz de valores
   const double&  probs[],     // matriz de probabilidades
   double&        quantile[]   // matriz para grabar los cuantiles
   )
```

Parámetros

array[]

[in] Matriz de valores.

probs[]

[in] Matriz de probabilidades.

quantile[]

[out] Matriz para grabar los cuantiles.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityEmpirical

La función calcula la densidad empírica de la probabilidad (pdf) para los valores aleatorios de la matriz.

```
bool  MathProbabilityDensityEmpirical(
   const double&  array[],   // matriz de valores aleatorios
   const int      count,     // número de parejas
   double&        x[],       // matriz de valores x
   double&        pdf[]      // matriz de valores pdf
   )
```

Parámetros

array[]

[in] matriz de valores aleatorios.

count

[in] Número de parejas (x, pdf(x)).

x[]

[out] Matriz para grabar los valores x.

pdf[]

[out] Matriz para grabar los valores pdf(x).

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionEmpirical

La función calcula la distribución empírica acumulativa(cdf) para los valores aleatorios de la matriz.

```
bool  MathCumulativeDistributionEmpirical(
   const double&  array[],   // matriz de valores aleatorios
   const int      count,     // número de parejas
   double&        x[],       // matriz de valores x
   double&        cdf[]      // matriz de valores cdf
   )
```

Parámetros

array[]

[in] matriz de valores aleatorios.

count

[in] Número de parejas (x, cdf(x)).

x[]

[out] Matriz para grabar los valores x.

cdf[]

[out] Matriz para grabar los valores cdf(x).

Valor devuelto

Devuelve true en caso de éxito, de lo contrario devuelve false.

© 2000-2025, MetaQuotes Ltd.

Fuzzy - biblioteca para trabajar con lógica difusa

La lógica difusa constituye una generalización de la lógica aristotélica tradicional cuando la verdad se ve como una variable lingüística. Como sucede con la lógica clásica, para la lógica difusa se encuentran definidas sus propias operaciones lógicas difusas sobre conjuntos difusos. Para los conjuntos difusos existen las mismas operaciones que para los conjuntos normales, excepto que sus cálculos son bastante más complejos. Destacaremos, asimismo, que la composición de los conjuntos difusos incluye un conjunto difuso.

Las peculiaridades principales de la lógica difusa que la diferencian de la clásica, son la proximidad máxima a la representación de la realidad y el alto nivel de subjetividad, debido a lo cual pueden surgir desviaciones significativas en los resultados de los cálculos con su uso.

Un modelo (o sistema) difuso es un modelo matemático en cuya base de cálculo se encuentra la lógica difusa. Se suele recurrir a este tipo de construcciones cuando el objeto de investigación tiene una formalización muy débil, y su descripción matemática exacta es demasiado compleja o desconocida. La calidad de los valores de salida de estos modelos (margen de error del modelo) depende directamente solo del experto que ha compuesto y ajustado el modelo. Para minimizar el error, la mejor opción será componer un modelo lo más completo y exhaustivo posible, configurándolo posteriormente mediante el aprendizaje de máquinas con una muestra amplia de aprendizaje.

El orden de construcción del modelo se puede dividir en tres etapas:

1. Definición de los parámetros de entrada y salida del modelo.
2. Construcción de la base de conocimiento del modelo.
3. Elección de uno de los métodos de deducción de lógica difusa (Mamdani o Sugeno).

Es precisamente de la primera etapa de la que dependen las otras dos, pues define el futuro funcionamiento del modelo. La base de conocimiento, o como se la llama también, la base de reglas es un conjunto de normas del tipo: "si, entonces", que define la relación mutua entre las entradas y salidas del objeto investigado. El número de reglas en el sistema no está limitado y es igualmente definido por el experto. El formato generalizado de reglas difusas es el siguiente:

Si se da la condición (envío) de la regla, entonces existe conclusión conforme a la regla.

La condición de la regla caracteriza el estado actual del objeto; la conclusión, por su parte, es la forma en que esta condición influye en el objeto. El aspecto general de las condiciones y conclusiones no se puede separar, puesto que es definido por una dedución de lógica difusa.

Cada regla en el sistema tiene su peso, este parámetro caracteriza el valor de la regla en el modelo. Los coeficientes de peso se asignan a la regla en el rango [0, 1]. En muchos ejemplos de modelos difusos que se pueden encontrar en la literatura especializada, los datos de los pesos no se indican, pero esto no significa que no existan, en realidad, en tal caso, el peso para cada regla de la base es fijo e igual a la unidad. Las condiciones y conclusiones de cada regla pueden ser de dos tipos:

1. simple, cuando en ella participa una variable difusa;
2. compuesto, cuando participan varias variables difusas.

Dependiendo de la base de conocimiento creada para el modelo, se definirá el sistema de dedución de lógica difusa. Se llama dedución de lógica difusa a la obtención de una conclusión en forma de conjunto difuso que corresponde a los valores actuales de las entradas usando una base difusa de conocimiento y operaciones difusas. Existen dos tipos principales de dedución de lógica difusa, Mamdani y Sugeno.

© 2000-2025, MetaQuotes Ltd.

Funciones de pertenencia

Se llama **función de pertenencia** *(membership function)* a la función que permite calcular el nivel de pertenencia de un elemento aleatorio de un conjunto universal a un conjunto difuso. Por consiguiente, la zona de valores de la función de pertenencia deberá entrar en el rango [0, 1].

En la mayoría de los casos, la función de pertenencia es monótona y continua.

Clase de funciones de pertenencia	Descripción
CConstantMembershipFunction	Clase para la implementación de una función de pertenencia en forma de recta paralela al eje de coordenadas.
CCompositeMembershipFunction	Clase para la implementación de la composición de funciones de pertenencia
CDifferencTwoSigmoidalMembershipFunction	Clase para la implementación de una función de pertenencia en forma de diferencia entre dos funciones sigmoidales con los parámetros A1, A2, C1, C2
CGeneralizedBellShapedMembershipFunction	Clase para la implementación de una función general de pertenencia en forma de campana con los parámetros A, B y C
CNormalCombinationMembershipFunction	Clase para la implementación de una función bilateral de pertenencia de Gauss con los parámetros B1, B2, Sigma1 y Sigma2
CNormalMembershipFunction	Clase para la implementación de una función simétrica de pertenencia de Gauss con los parámetros B y Sigma
CP_ShapedMembershipFunction	Clase para la implementación de una función de pertenencia en forma de Pi con los parámetros A, B, C y D
CProductTwoSigmoidalMembershipFunction	Clase para la implementación de una función de pertenencia en forma de producto de dos funciones sigmoidales con los parámetros A1, A2, C1, C2
CS_ShapedMembershipFunction	Clase para la implementación de una función de pertenencia en forma de S con los parámetros A y B
CTrapezoidMembershipFunction	Clase para la implementación de una función de pertenencia en forma de trapecio con los parámetros X1, X2, X3 y X4
CTriangularMembershipFunction	Clase para la implementación de una función de pertenencia en forma de triángulo con los parámetros X1, X2, X3

© 2000-2025, MetaQuotes Ltd.

Clase de funciones de pertenencia	Descripción
CSigmoidalMembershipFunction	Clase para la implementación de una función sigmoidal de pertenencia con los parámetros A y C.
CZ_ShapedMembershipFunction	Clase para la implementación de una función de pertenencia en forma de Z con los parámetros A y B.
IMembershipFunction	Clase básica para todas las clases de las funciones de pertenencia.

© 2000-2025, MetaQuotes Ltd.

CConstantMembershipFunction

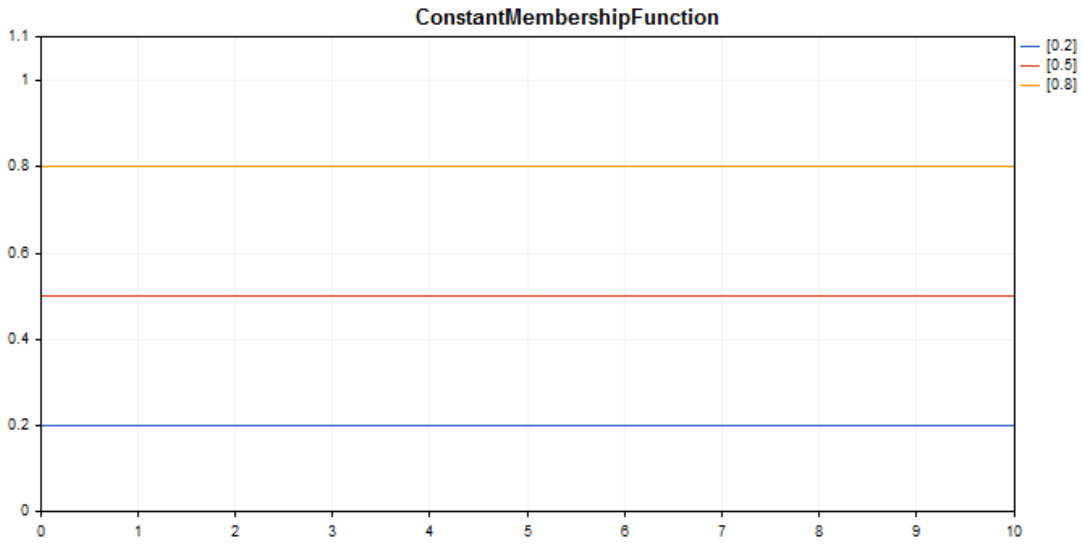
Clase para la implementación de la función de pertenencia en forma de recta paralela al eje de coordenadas.

Descripción

La función se describe con la ecuación:

y(x)=c

Por consiguiente, el nivel de pertenencia para esta función es el mismo en todo el eje numérico y se iguala al parámetro indicado en el constructor.



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CConstantMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject
IMembershipFunction
CConstantMembershipFunction

Métodos de clase

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                                  ConstantMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CConstantMembershipFunction func1(0.2);
CConstantMembershipFunction func2(0.5);
CConstantMembershipFunction func3(0.8);
//--- Create wrappers for membership functions
double ConstantMembershipFunction1(double x) { return(func1.GetValue(x)); }
double ConstantMembershipFunction2(double x) { return(func2.GetValue(x)); }
double ConstantMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"ConstantMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"ConstantMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ConstantMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(ConstantMembershipFunction1,0.0,10.0,1.0,CURVE_LINES,"[0.2]");
   graphic.CurveAdd(ConstantMembershipFunction2,0.0,10.0,1.0,CURVE_LINES,"[0.5]");
   graphic.CurveAdd(ConstantMembershipFunction3,0.0,10.0,1.0,CURVE_LINES,"[0.8]");
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const double  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

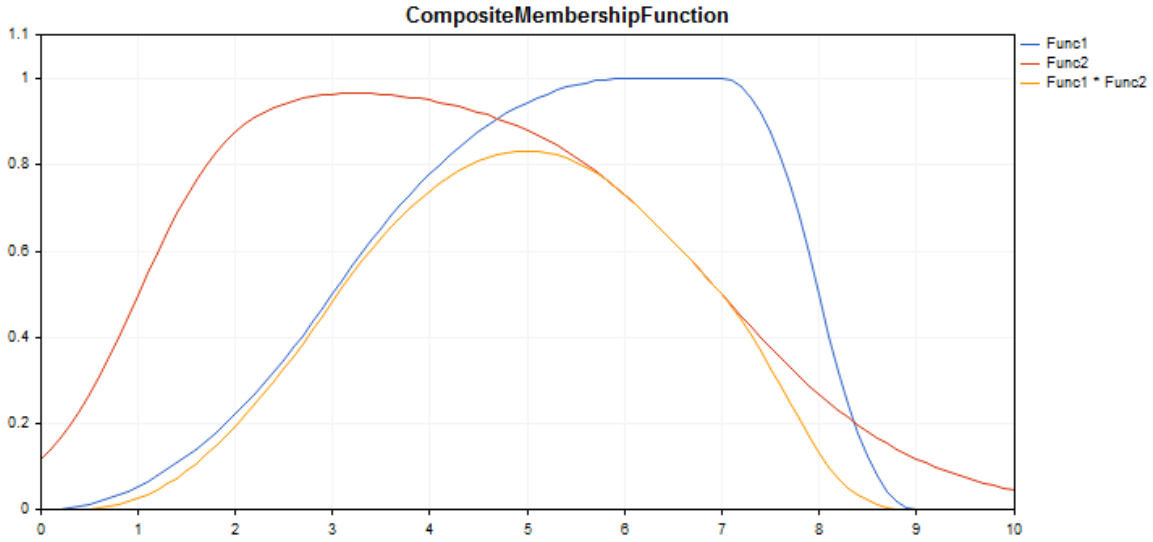
© 2000-2025, MetaQuotes Ltd.

CCompositeMembershipFunction

Clase para la implementación de la composición de funciones de pertenencia.

Descripción

La composición de las funciones de pertenencia es la unión de dos o más funciones de pertenencia con la ayuda de un operador establecido.



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CCompositeMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject
  IMembershipFunction
    CCompositeMembershipFunction

Métodos de clase

Método de clase	Descripción
CompositionType	Establece el operador de composición.
MembershipFunctions	Retorna la lista de funciones de pertenencia.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                                  CompositeMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CP_ShapedMembershipFunction func2(0,6,7,9);
CCompositeMembershipFunction composite(ProdMF,GetPointer(func1),GetPointer(func2));
//--- Create wrappers for membership functions
double ProductTwoSigmoidalMembershipFunctions(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction(double x) { return(func2.GetValue(x)); }
double CompositeMembershipFunction(double x) { return(composite.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"CompositeMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"CompositeMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("CompositeMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(P_ShapedMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1");
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions,0.0,10.0,0.1,CURVE_LINES,"F
   graphic.CurveAdd(CompositeMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1 * Func
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

CompositionType

Establece el operador de composición.

```
void  CompositionType(
   MfCompositionType  value      // tipo de operador
   )
```

Parámetros

value

[in] Tipo de operador de composición.

Nota

Están disponibles los siguientes tipos de operador:

- MinMF (mínimo de funciones)
- MaxMF (máximo de funciones)
- ProdMF (producto de funciones)
- SumMF (suma de funciones)

MembershipFunctions

Retorna la lista de las funciones de pertenencia que entran en la composición

```
CList*  MembershipFunctions(
   void        // lista de las funciones de pertenencia
   )
```

Valor devuelto

Lista de las funciones de pertenencia.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

© 2000-2025, MetaQuotes Ltd.

Valor de la función de pertenencia.

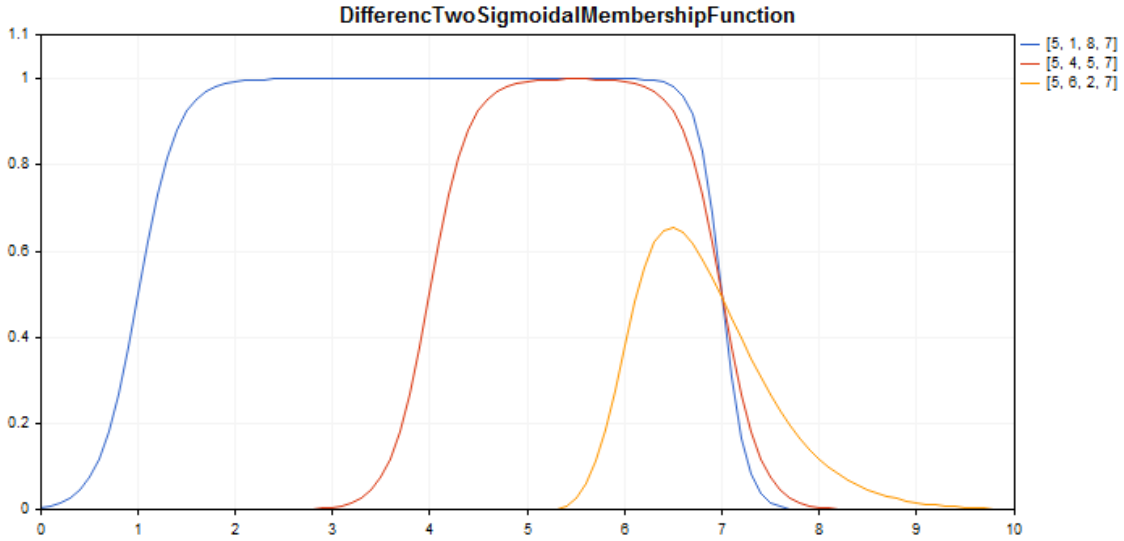
© 2000-2025, MetaQuotes Ltd.

CDifferencTwoSigmoidalMembershipFunction

Clase para la implementación de la función de pertenencia en forma de diferencia entre dos funciones sigmoidales con los parámetros A1, A2, C1, C2.

Descripción

La función se basa en el uso de una curva sgmoidal. Con su ayuda se puede crear una función de pertenencia con valores iguales a 1, comenzando por un cierto valor del argumento. Tales funciones convienen si es necesario indicar términos lingüísticos del tipo "corto" o "largo".



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CDifferencTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject

IMembershipFunction

CDifferencTwoSigmoidalMembershipFunction

Métodos de clase

Método de clase	Descripción
A1	Retorna y establece el coeficiente de inclinación de la primera función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
A2	Retorna y establece el coeficiente de inclinación de la segunda función de pertenencia.
C1	Retorna y establece el parámetro de la coordenada de inflexión de la primera función de pertenencia.
C2	Retorna y establece el parámetro de la coordenada de inflexión de la segunda función de pertenencia.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                      DifferencTwoSigmoidalMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd.     |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CDifferencTwoSigmoidalMembershipFunction func1(5,1,8,7);
CDifferencTwoSigmoidalMembershipFunction func2(5,4,5,7);
CDifferencTwoSigmoidalMembershipFunction func3(5,6,2,7);
//--- Create wrappers for membership functions
double DifferencTwoSigmoidalMembershipFunction1(double x) { return(func1.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction2(double x) { return(func2.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"DifferencTwoSigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"DifferencTwoSigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("DifferencTwoSigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (Método Get)

Retorna el coeficiente de inclinación de la primera función de pertenencia.

```
double  A1()
```

Valor devuelto

Valor del coeficiente de inclinación.

A1 (Método Set)

Establece el coeficiente de inclinación de la primera función de pertenencia.

```
void  A1(
   const double  a1      // valor del coeficiente de inclinación
   )
```

Parámetros

a1

[in] Valor del coeficiente de inclinación.

A2 (Método Get)

Retorna el coeficiente de inclinación de la segunda función de pertenencia.

```
double  A2()
```

Valor devuelto

Valor del coeficiente de inclinación.

A2 (Método Set)

© 2000-2025, MetaQuotes Ltd.

Establece el coeficiente de inclinación de la segunda función de pertenencia.

```
void  A2(
   const double  a2      // valor del coeficiente de inclinación
   )
```

Parámetros

a2

[in] Valor del coeficiente de inclinación.

C1 (Método Get)

Retorna el parámetro de la coordenada de inflexión de la primera función de pertenencia.

```
double  C1()
```

Valor devuelto

Valor de la coordenada de inflexión.

C1 (Método Set)

Establece el parámetro de la coordenada de inflexión de la primera función de pertenencia.

```
void  C1(
   const double  c1      // valor de la coordenada de inflexión
   )
```

Parámetros

c1

[in] Valor de la coordenada de inflexión.

C2 (Método Get)

Retorna el parámetro de la coordenada de inflexión de la segunda función de pertenencia.

```
double  C2()
```

Valor devuelto

Valor de la coordenada de inflexión.

C2 (Método Set)

Establece el parámetro de la coordenada de inflexión de la segunda función de pertenencia.

```
void  C2(
   const double  c2      // valor de la coordenada de inflexión
   )
```

© 2000-2025, MetaQuotes Ltd.

Parámetros

c2

[in] Valor de la coordenada de inflexión.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const double  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

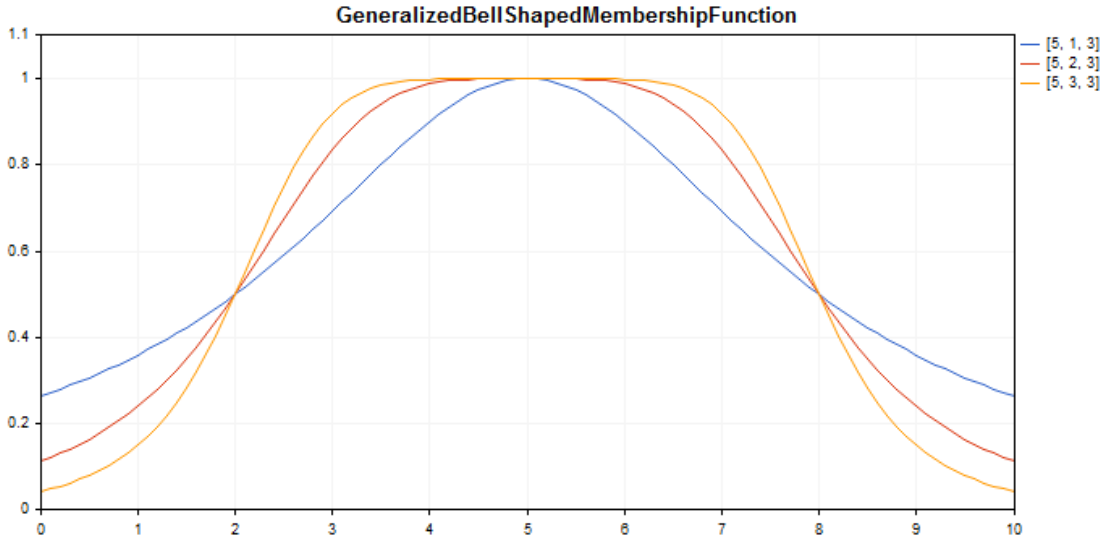
© 2000-2025, MetaQuotes Ltd.

CGeneralizedBellShapedMembershipFunction

Clase para la implementación de la función general de pertenencia en forma de campana con los parámetros A, B y C.

Descripción

La función generalizada de pertenencia en forma de campana es parecida por su forma a la de Gauss. En toda la zona de definición la función es suave y adopta valores diferentes a cero.



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CGeneralizedBellShapedMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject

IMembershipFunction

CGeneralizedBellShapedMembershipFunction

Métodos de clase

Método de clase	Descripción
A	Retorna y establece el coeficiente de concentración de la función de pertenencia.
B	Retorna y establece el coeficiente de inclinación de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
C	Retorna y establece la coordenada del máximo de la función de pertenencia.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                      GeneralizedBellShapedMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CGeneralizedBellShapedMembershipFunction func1(5, 1, 3);
CGeneralizedBellShapedMembershipFunction func2(5, 2, 3);
CGeneralizedBellShapedMembershipFunction func3(5, 3, 3);
//--- Create wrappers for membership functions
double GeneralizedBellShapedMembershipFunction1(double x) { return(func1.GetValue(x));
double GeneralizedBellShapedMembershipFunction2(double x) { return(func2.GetValue(x));
double GeneralizedBellShapedMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"GeneralizedBellShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"GeneralizedBellShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("GeneralizedBellShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Método Get)

Retorna el coeficiente de concentración de la función de pertenencia.

```
double  A()
```

Valor devuelto

Valor del coeficiente de concentración.

A (Método Set)

Establece el coeficiente de concentración de la función de pertenencia.

```
void  A(
   const double  a      // valor del coeficiente de concentración
   )
```

Parámetros

a

[in] Coeficiente de concentración de la función de pertenencia.

B (Método Get)

Retorna el coeficiente de inclinación de la primera función de pertenencia.

```
double  B()
```

Valor devuelto

Valor del coeficiente de inclinación.

B (Método Set)

Establece el coeficiente de inclinación de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

```
void  B(
   const double  b      // valor del coeficiente de inclinación
   )
```

Parámetros

b

[in] Coeficiente de inclinación de la función de pertenencia.

C (Método Get)

Retorna la coordenada del máximo de la función de pertenencia.

```
double  C()
```

Valor devuelto

Coordenada del máximo de la función de pertenencia.

C (Método Set)

Establece la coordenada del máximo de la función de pertenencia.

```
void  C(
   const double  c      // valor de la coordenada del máximo
   )
```

Parámetros

c

[in] Coordenada del máximo de la función de pertenencia.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

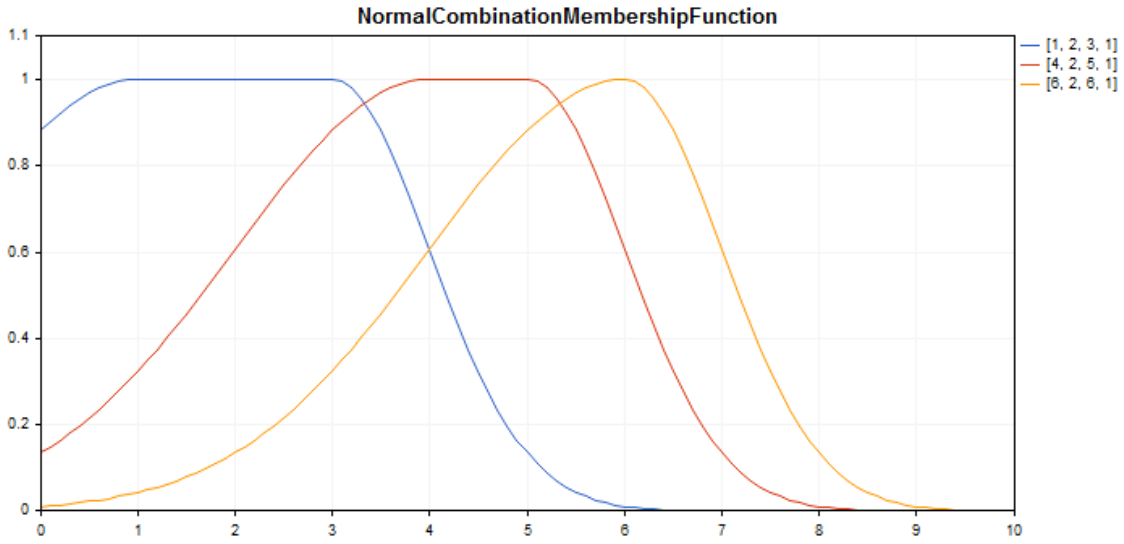
© 2000-2025, MetaQuotes Ltd.

CNormalCombinationMembershipFunction

Clase para la implementación de la función bilateral de pertenencia de Gauss con los parámetros A, B y C.

Descripción

La función bilateral de pertenencia de Gauss se forma usando la distribución de Gauss. Permite establecer una función de pertenencia asimétrica. En toda la zona de definición la función es suave y adopta valores diferentes a cero.



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CNormalCombinationMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject

IMembershipFunction

CNormalCombinationMembershipFunction

Métodos de clase

Método de clase	Descripción
B1	Retorna y establece el valor del primer centro de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
B2	Retorna y establece el valor del segundo centro de la función de pertenencia.
Sigma1	Retorna y establece el primer parámetro de curvatura de la función de pertenencia.
Sigma2	Retorna y establece el segundo parámetro de curvatura de la función de pertenencia.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                          NormalCombinationMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CNormalCombinationMembershipFunction func1(1,2,3,1);
CNormalCombinationMembershipFunction func2(4,2,5,1);
CNormalCombinationMembershipFunction func3(6,2,6,1);
//--- Create wrappers for membership functions
double NormalCombinationMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalCombinationMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalCombinationMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"NormalCombinationMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalCombinationMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalCombinationMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(NormalCombinationMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[1,
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurveAdd(NormalCombinationMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[4,
   graphic.CurveAdd(NormalCombinationMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[6,
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B1 (Método Get)

Retorna el valor del primer centro de la función de pertenencia.

```
double  B1()
```

Valor devuelto

Valor del primer centro de la función de pertenencia.

B1 (Método Set)

Establece el valor del primer centro de la función de pertenencia.

```
void  B1(
   const double  b1      // valor del primer centro
   )
```

Parámetros

b

[in] Valor del primer centro de la función de pertenencia.

B2 (Método Get)

Retorna el valor del segundo centro de la función de pertenencia.

```
double  B2()
```

Valor devuelto

Valor del segundo centro de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

B2 (Método Set)

Establece el valor del segundo centro de la función de pertenencia.

```
void  B2(
   const double  b2      // valor del segundo centro
   )
```

Parámetros

b2

[in] Valor del segundo centro de la función de pertenencia.

Sigma1 (Método Get)

Retorna el primer parámetro de curvatura de la función de pertenencia.

```
double  Sigma1()
```

Valor devuelto

Valor del primer parámetro de curvatura de la función de pertenencia.

Sigma1 (Método Set)

Establece el valor del primer parámetro de curvatura de la función de pertenencia.

```
void  Sigma1(
   const double  sigma1      // valor del primer parámetro de curvatura
   )
```

Parámetros

sigma1

[in] Primer parámetro de curvatura de la función de pertenencia.

Sigma2 (Método Get)

Retorna el segundo parámetro de curvatura de la función de pertenencia.

```
double  Sigma2()
```

Valor devuelto

Valor del segundo parámetro de curvatura de la función de pertenencia.

Sigma2 (Método Set)

© 2000-2025, MetaQuotes Ltd.

Establece el valor del segundo parámetro de curvatura de la función de pertenencia.

```
void  Sigma2(
   const double  sigma2      // valor del segundo parámetro de curvatura
   )
```

Parámetros

sigma2

[in] Segundo parámetro de curvatura de la función de pertenencia.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

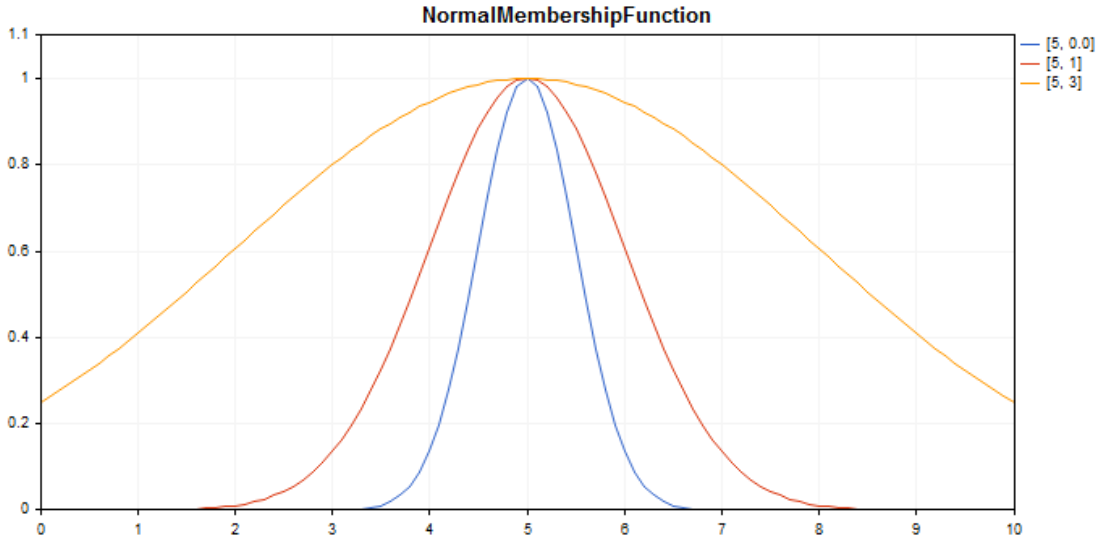
© 2000-2025, MetaQuotes Ltd.

CNormalMembershipFunction

Clase para la implementación de la función simétrica de pertenencia de Gauss con los parámetros B y Sigma.

Descripción

La función simétrica de pertenencia de Gauss se forma usando la distribución de Gauss. En toda la zona de definición la función es suave y adopta valores diferentes a cero.



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CNormalMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject
IMembershipFunction
CNormalMembershipFunction

Métodos de clase

Método de clase	Descripción
B	Retorna y establece el centro de la función de pertenencia.
Sigma	Retorna y establece el parámetro de curvatura de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                                     NormalMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CNormalMembershipFunction func1(5,0.5);
CNormalMembershipFunction func2(5,1);
CNormalMembershipFunction func3(5,3);
//--- Create wrappers for membership functions
double NormalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"NormalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(NormalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[5, 0.0]");
   graphic.CurveAdd(NormalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[5, 1]");
   graphic.CurveAdd(NormalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[5, 3]");
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B (Método Get)

Retorna el centro de la función de pertenencia.

```
double  B()
```

Valor devuelto

Valor del centro de la función de pertenencia.

B (Método Set)

Establece el valor del centro de la función de pertenencia.

```
void  B(
   const double  b      // valor del centro de la función
   )
```

Parámetros

b

[in] Valor del centro de la función de pertenencia.

Sigma (Método Get)

Retorna el parámetro de curvatura de la función de pertenencia

```
double  Sigma()
```

Valor devuelto

Valor del parámetro de curvatura de la función de pertenencia.

Sigma (Método Set)

Establece el valor del parámetro de curvatura de la función de pertenencia.

```
void  Sigma(
   const double  sigma      // valor del parámetro de curvatura
   )
```

Parámetros

sigma

© 2000-2025, MetaQuotes Ltd.

[in] Parámetro de curvatura de la función de pertenencia.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const double  x      // argumento
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

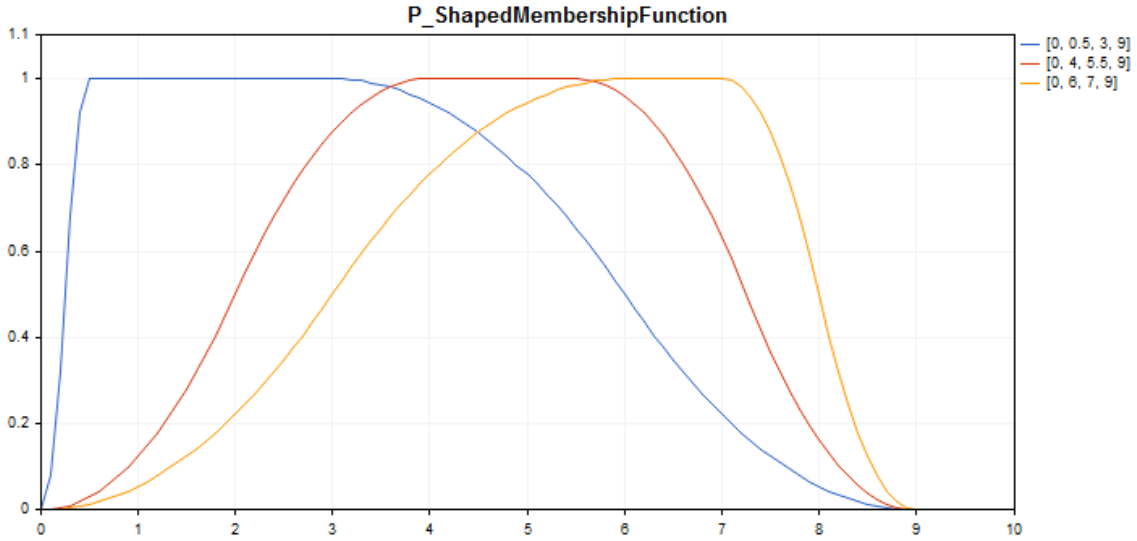
© 2000-2025, MetaQuotes Ltd.

CP_ShapedMembershipFunction

Clase para la implementación de la función de pertenencia en forma de Pi con los parámetros A, B, C y D.

Descripción

La función de pertenencia en forma de Pi tiene la forma de un trapecio de líneas curvas. La función se usa para crear funciones de pertenencia asimétricas con una transición suave de una valoración pesimista de un número difuso hacia una valoración optimista del mismo.



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CP_ShapedMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject

IMembershipFunction

CP_ShapedMembershipFunction

Métodos de clase

Método de clase	Descripción
A	Retorna y establece el parámetro del comienzo de un conjunto difuso.

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
B	Retorna y establece el primer parámetro del núcleo de un conjunto difuso.
C	Retorna y establece el segundo parámetro del núcleo de un conjunto difuso.
D	Retorna y establece el parámetro de la finalización de un conjunto difuso.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                                    P_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CP_ShapedMembershipFunction func1(0,0.5,3,9);
CP_ShapedMembershipFunction func2(0,4,5.5,9);
CP_ShapedMembershipFunction func3(0,6,7,9);
//--- Create wrappers for membership functions
double P_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double P_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"P_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"P_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("P_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(P_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 0.5, 3,
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurveAdd(P_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 4, 5.5,
   graphic.CurveAdd(P_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[0, 6, 7, 9]
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Método Get)

Retorna el comienzo de un conjunto difuso.

```
double  A()
```

Valor devuelto

Parámetro de comienzo de un conjunto difuso.

A (Método Set)

Establece el parámetro del comienzo de un conjunto difuso.

```
void  A(
   const double  a      // parámetro del comienzo de un conjunto difuso
   )
```

Parámetros

a

[in] Parámetro del comienzo de un conjunto difuso.

B (Método Get)

Retorna el primer parámetro del núcleo de un conjunto difuso.

```
double  B()
```

Valor devuelto

Primer parámetro del núcleo de un conjunto difuso.

B (Método Set)

© 2000-2025, MetaQuotes Ltd.

Establece el primer parámetro del núcleo de un conjunto difuso.

```
void  B(
   const double  b      // valor del primer parámetro del núcleo de un conjunto difuso
   )
```

Parámetros

b

[in] Primer parámetro del núcleo de un conjunto difuso.

C (Método Get)

Retorna el segundo parámetro del núcleo de un conjunto difuso.

```
double  C()
```

Valor devuelto

Segundo parámetro del núcleo de un conjunto difuso.

C (Método Set)

Establece el segundo parámetro del núcleo de un conjunto difuso.

```
void  C(
   const double  c      // valor del segundo parámetro del núcleo de un conjunto difuso
   )
```

Parámetros

c

[in] Segundo parámetro del núcleo de un conjunto difuso.

D (Método Get)

Retorna el parámetro de la finalización de un conjunto difuso.

```
double  D()
```

Valor devuelto

Parámetro de la finalización de un conjunto difuso.

D (Método Set)

Establece el parámetro de la finalización de un conjunto difuso.

```
void  D(
   const double  d      // valor del parámetro de la finalización de un conjunto difuso
   )
```

Parámetros

d

© 2000-2025, MetaQuotes Ltd.

[in] Valor del parámetro de la finalización de un conjunto difuso.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado

```
double  GetValue(
   const double  x
   )
```

Parámetros

x

[in] argumento de la función de pertenencia

Valor devuelto

valor de la función de pertenencia

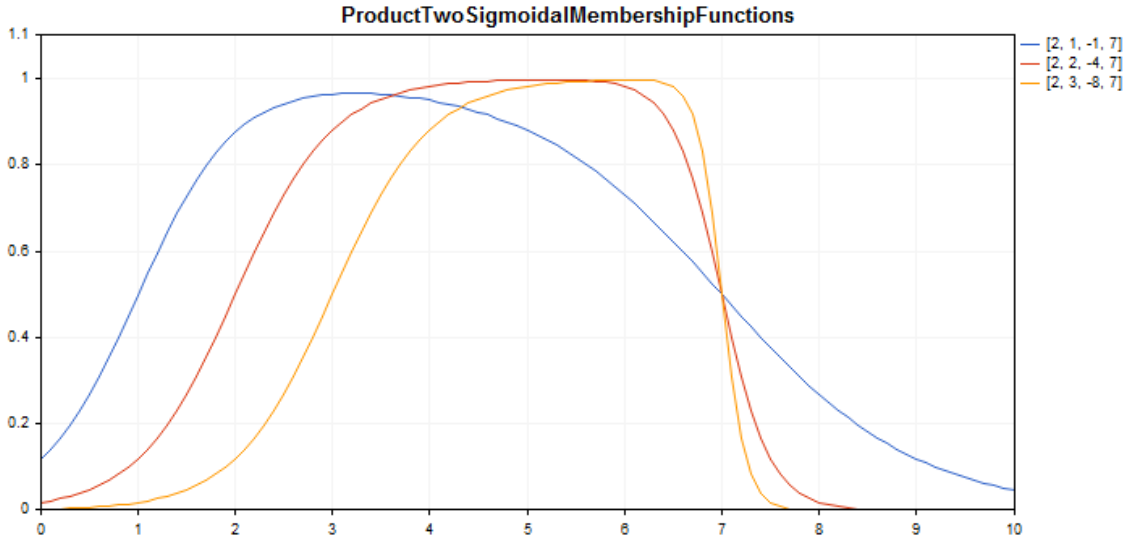
© 2000-2025, MetaQuotes Ltd.

CProductTwoSigmoidalMembershipFunction

Clase para la implementación de la función de pertenencia en forma de producto de dos funciones sigmoidales con los parámetros A1, A2, C1, C2.

Descripción

El producto de funciones de pertenenecia sigmoidales se aplica para crear funciones asimétricas suaves. Con su ayuda se puede crear una función de pertenencia con valores iguales a 1, comenzando por un cierto valor del argumento. Tales funciones convienen si es necesario indicar términos lingüísticos del tipo "corto" o "largo".



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CProductTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject

IMembershipFunction

CProductTwoSigmoidalMembershipFunctions

Métodos de clase

Método de clase	Descripción
A1	Retorna y establece el coeficiente de inclinación de la primera función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
A2	Retorna y establece el coeficiente de inclinación de la segunda función de pertenencia.
C1	Retorna el parámetro de la coordenada de inflexión de la primera función de pertenencia.
C2	Retorna el parámetro de la coordenada de inflexión de la segunda función de pertenencia.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                      ProductTwoSigmoidalMembershipFunctions.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CProductTwoSigmoidalMembershipFunctions func2(2,2,-4,7);
CProductTwoSigmoidalMembershipFunctions func3(2,3,-8,7);
//--- Create wrappers for membership functions
double ProductTwoSigmoidalMembershipFunctions1(double x) { return(func1.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions2(double x) { return(func2.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"ProductTwoSigmoidalMembershipFunctions",0,30,30,780,380))
     {
      graphic.Attach(0,"ProductTwoSigmoidalMembershipFunctions");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ProductTwoSigmoidalMembershipFunctions");
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions1,0.0,10.0,0.1,CURVE_LINES,"
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions2,0.0,10.0,0.1,CURVE_LINES,"
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions3,0.0,10.0,0.1,CURVE_LINES,"
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (Método Get)

Retorna el coeficiente de inclinación de la primera función de pertenencia.

```
double  A1()
```

Valor devuelto

Coeficiente de inclinación de la primera función de pertenencia.

A1 (Método Set)

Establece el coeficiente de inclinación de la primera función de pertenencia.

```
void  A1(
   const double  a1      // coeficiente de inclinación de la primera función de perter
   )
```

Parámetros

a1

[in] Coeficiente de inclinación de la primera función de pertenencia.

A2 (Método Get)

Retorna el coeficiente de inclinación de la segunda función de pertenencia.

```
double  A2()
```

Valor devuelto

Coeficiente de inclinación de la segunda función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

A2 (Método Set)

Establece el coeficiente de inclinación de la segunda función de pertenencia.

```
void  A2(
   const double  a2      // coeficiente de inclinación de la segunda función de perter
   )
```

Parámetros

a2

[in] Coeficiente de inclinación de la segunda función de pertenencia.

C1 (Métod Get)

Retorna el parámetro de la coordenada de inflexión de la primera función de pertenencia.

```
double  C1()
```

Valor devuelto

Coordenada de inflexión de la primera función de pertenencia.

C1 (Método Set)

Establece la coordenada de inflexión de la primera función de pertenencia.

```
void  C1(
   const double  c1      // coordenada de inflexión de la primera función de pertenenc
   )
```

Parámetros

c1

[in] Coordenada de inflexión de la primera función de pertenencia.

C2 (Método Get)

Retorna el parámetro de la coordenada de inflexión de la segunda función de pertenencia.

```
double  C2()
```

Valor devuelto

Coordenada de inflexión de la segunda función de pertenencia.

C2 (Método Set)

Establece la coordenada de inflexión de la segunda función de pertenencia.

```
void  C2(
   const double  c2      // coordenada de inflexión de la segunda función de pertenenc
   )
```

© 2000-2025, MetaQuotes Ltd.

Parámetros

c2

[in] Coordenada de inflexión de la segunda función de pertenencia.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

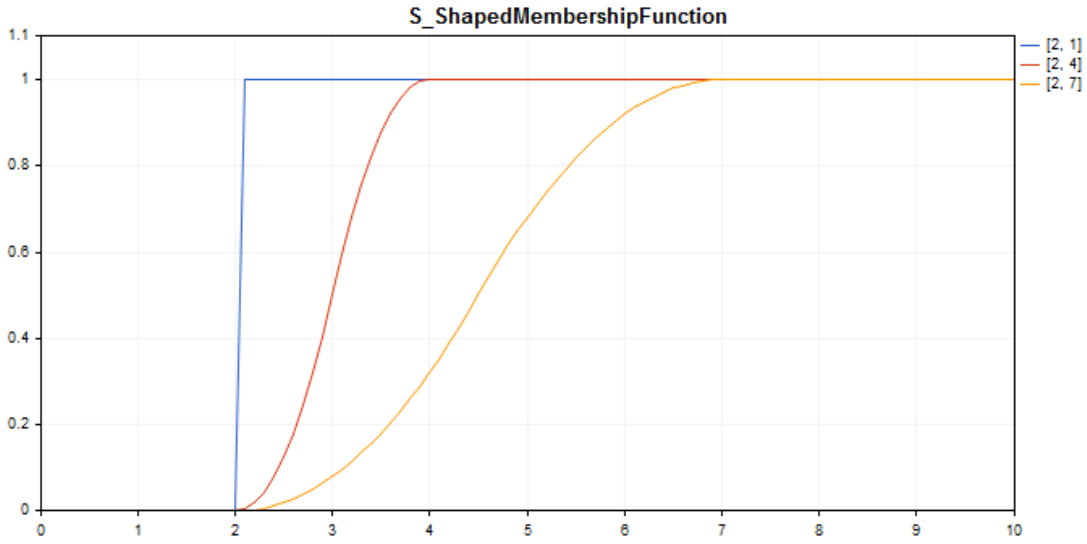
CS_ShapedMembershipFunction

Clase para la implementación de la función de pertenencia en forma de S con los parámetros A y B.

Descripción

La función crea una función de pertenencia en forma de S de dos parámetros. Se trata de una función no disminutiva que adopta valores de 0 a 1. Los parámetros A y B definen el intervalo dentro del cual la función crece en trayectoria lineal de 0 a 1.

Con la ayuda de esta función se presentan los conjuntos difusos del tipo "muy alto" (es decir, se crean funciones de pertenencia no decrecientes con saturación).



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CS_ShapedMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject
  IMembershipFunction
    CS_ShapedMembershipFunction

Métodos de clase

Método de clase	Descripción
A	Retorna y establece el parámetro del comienzo del intervalo de incremento.

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
B	Retorna y establece el primer parámetro del núcleo de un conjunto difuso.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                                  S_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CS_ShapedMembershipFunction func1(2,1);
CS_ShapedMembershipFunction func2(2,4);
CS_ShapedMembershipFunction func3(2,7);
//--- Create wrappers for membership functions
double S_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double S_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double S_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"S_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"S_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("S_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(S_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(S_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
   graphic.CurveAdd(S_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 7]");
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Método Get)

Retorna el parámetro del comienzo del intervalo de incremento.

```
double  A()
```

Valor devuelto

Parámetro del comienzo del intervalo de incremento.

A (Método Set)

Establece el parámetro del comienzo del intervalo de incremento.

```
void  A(
   const double  a      // parámetro del comienzo del intervalo de incremento
   )
```

Parámetros

a

[in] Parámetro del comienzo del intervalo de incremento.

B (Método Get)

Retorna el primer parámetro del núcleo de un conjunto difuso.

```
double  B()
```

Valor devuelto

Primer parámetro del núcleo de un conjunto difuso.

B (Método Set)

Establece el primer parámetro del núcleo de un conjunto difuso.

```
void  B(
   const double  b      // valor del primer parámetro del núcleo de un conjunto difuso
   )
```

© 2000-2025, MetaQuotes Ltd.

Parámetros

b

[in] Primer parámetro del núcleo de un conjunto difuso.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

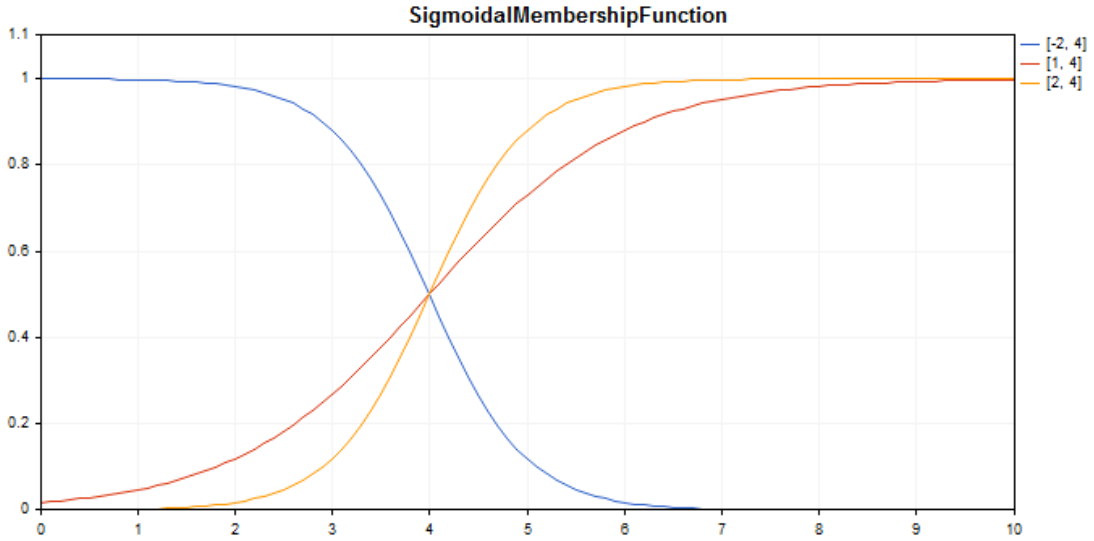
© 2000-2025, MetaQuotes Ltd.

CSigmoidalMembershipFunction

Clase para la implementación de la función sigmoidal de pertenencia con los parámetros A y C.

Descripción

La función sigmoidal se usa para establecer funciones de pertenencia monótonas. Con su ayuda se puede crear una función de pertenencia con valores iguales a 1, comenzando por un cierto valor del argumento. Tales funciones convienen si es necesario indicar términos lingüísticos del tipo "corto" o "largo".



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CSigmoidalMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject

IMembershipFunction

CSigmoidalMembershipFunction

Métodos de clase

Método de clase	Descripción
A	Retorna y establece el coeficiente de inclinación de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
C	Retorna y establece el parámetro de la coordenada de inflexión de la función de pertenencia.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                                  SigmoidalMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CSigmoidalMembershipFunction func1(-2, 4);
CSigmoidalMembershipFunction func2(1, 4);
CSigmoidalMembershipFunction func3(2, 4);
//--- Create wrappers for membership functions
double SigmoidalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double SigmoidalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double SigmoidalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"SigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"SigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("SigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(SigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[-2, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[1, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Método Get)

Retorna el coeficiente de inclinación de la primera función de pertenencia.

```
double  A()
```

Valor devuelto

Coeficiente de inclinación de la función de pertenencia.

A (Método Set)

Establece el coeficiente de inclinación de la función de pertenencia.

```
void  A(
   const double  a      // coeficiente de inclinación de la función de pertenencia
   )
```

Parámetros

a

[in] Coeficiente de inclinación de la función de pertenencia.

C (Métofo Get)

Retorna el parámetro de la coordenada de inflexión de la función de pertenencia.

```
double  C()
```

Valor devuelto

Coordenada de inflexión de la función de pertenencia.

C (Método Set)

Establece la coordenada de inflexión de la función de pertenencia.

```
void  C(
   const double  c      // coordenada de inflexión de la función de pertenencia
   )
```

© 2000-2025, MetaQuotes Ltd.

Parámetros

c

[in] Coordenada de inflexión de la función de pertenencia.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

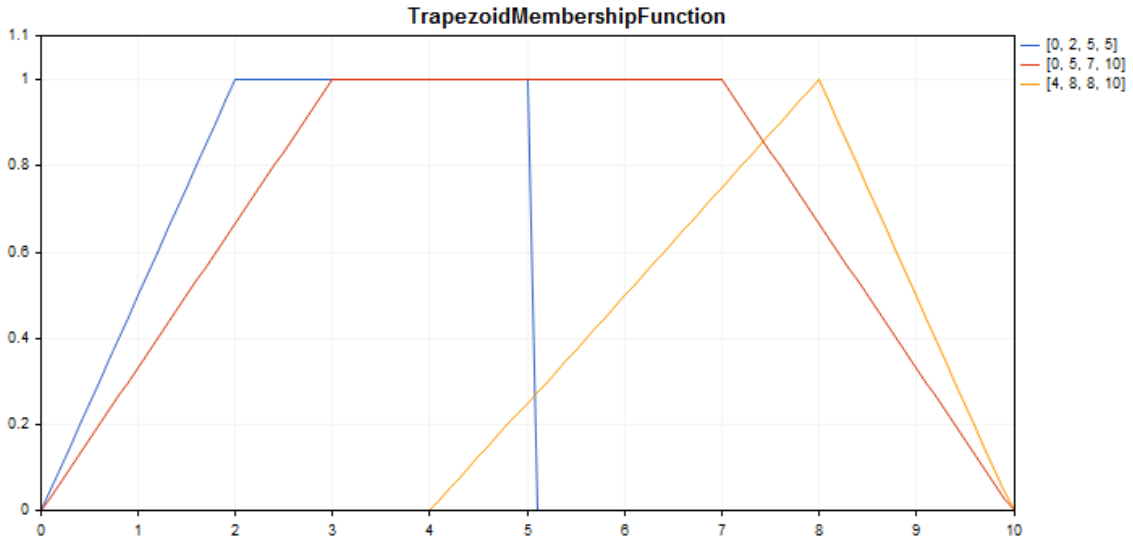
CTrapezoidMembershipFunction

Clase para la implementación de la función de pertenencia en forma de trapecio con los parámetros X1, X2, X3 y X4.

Descripción

La función se forma con el uso de una aproximación lineal por segmentos. Se trata de una generalización de la función triangular, que permite establecer el núcleo de un conjunto difuso en forma de intervalo. Esta función de pertenencia da la posibilidad de interpretar cómodamente la valoración optmista/pesimista.

Se usa para definir funciones asimétricas de pertenencia de variables, cuyos valores más probables se aplican en un cierto intervalo.



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CTrapezoidMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject

IMembershipFunction

CTrapezoidMembershipFunction

Métodos de clase

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
X1	Retorna y establece el valor del primer punto en la abscisa.
X2	Retorna y establece el valor del segundo punto en la abscisa.
X3	Retorna y establece el valor del tercer punto en la abscisa.
X4	Retorna y establece el valor del cuarto punto en la abscisa.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                                  TrapezoidMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CTrapezoidMembershipFunction func1(0,2,5,5);
CTrapezoidMembershipFunction func2(0,3,7,10);
CTrapezoidMembershipFunction func3(4,8,8,10);
//--- Create wrappers for membership functions
double TrapezoidMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TrapezoidMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TrapezoidMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"TrapezoidMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TrapezoidMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TrapezoidMembershipFunction");
   graphic.BackgroundMainSize(16);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- create curve
   graphic.CurveAdd(TrapezoidMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5, 5
   graphic.CurveAdd(TrapezoidMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 7, 1
   graphic.CurveAdd(TrapezoidMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[4, 8, 8, 1
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (Método Get)

Retorna el valor del primer punto en la abscisa.

```
double  X1()
```

Valor devuelto

Valor del primer punto en la abscisa.

X1 (Método Set)

Establece el valor del primer punto en la abscisa.

```
void  X1(
   const double  x1      // valor del primer punto en la abscisa
   )
```

Parámetros

x1

[in] Valor del primer punto en la abscisa.

X2 (Método Get)

Retorna valor del segundo punto en la abscisa.

```
double  X2()
```

Valor devuelto

Valor del segundo punto en la abscisa.

© 2000-2025, MetaQuotes Ltd.

X2 (Método Set)

Establece el valor del segundo punto en la abscisa.

```
void  X2(
   const double  x2      // valor del segundo punto en la abscisa
   )
```

Parámetros

x2

[in] Valor del segundo punto en la abscisa.

X3 (Método Get)

Retorna el valor del tercer punto en la abscisa.

```
double  X3()
```

Valor devuelto

Valor del tercer punto en la abscisa.

X3 (Método Set)

Establece el valor del tercer punto en la abscisa.

```
void  X3(
   const double  x3      // valor del tercer punto en la abscisa
   )
```

Parámetros

x3

[in] Valor del tercer punto en la abscisa.

X4 (Método Get)

Retorna el valor del cuarto punto en la abscisa.

```
double  X4()
```

Valor devuelto

Valor del cuarto punto en la abscisa.

X4 (Método Set)

Establece el valor del cuarto punto en la abscisa.

```
void  X4(
   const double  x4      // valor del cuarto punto en la abscisa
   )
```

© 2000-2025, MetaQuotes Ltd.

Parámetros

x4

[in] Valor del cuarto punto en la abscisa.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

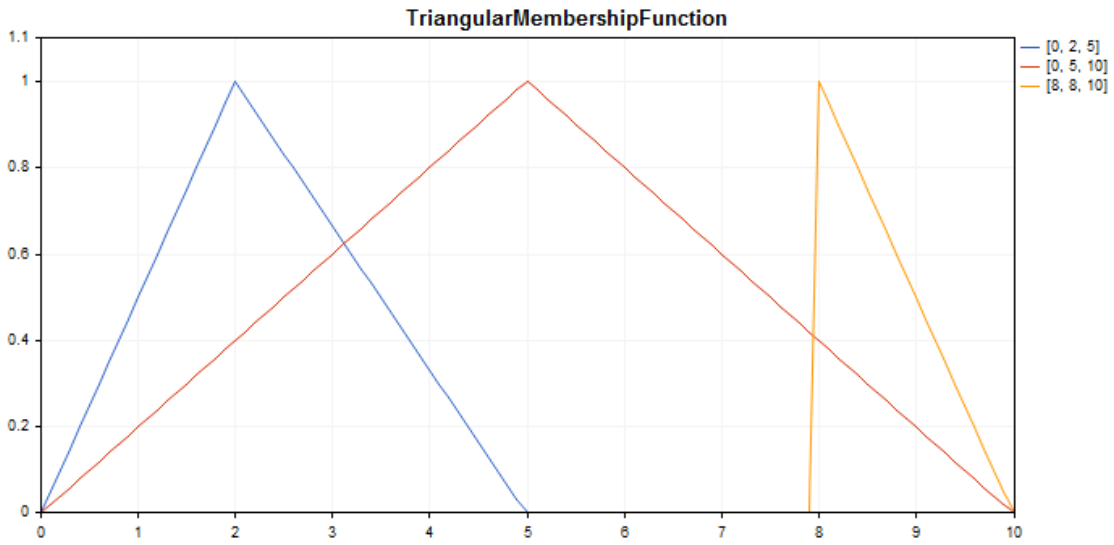
© 2000-2025, MetaQuotes Ltd.

CTriangularMembershipFunction

Clase para la implementación de la función de pertenencia en forma de triángulo con los parámetros X1, X2, X3 y X4.

Descripción

La función crea una función de pertenencia en forma de triángulo. Se trata de una función de pertenencia sencilla, además de ser la más usada.



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CTriangularMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject

IMembershipFunction

CTriangularMembershipFunction

Métodos de clase

Método de clase	Descripción
X1	Retorna el valor del primer punto en la abscisa.
X2	Retorna valor del segundo punto en la abscisa.
X3	Retorna el valor del tercer punto en la abscisa.

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
ToNormalMF	Transforma la función triangular de pertenencia en una función de Gauss.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                                TriangularMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CTriangularMembershipFunction func1(0,2,5);
CTriangularMembershipFunction func2(0,5,10);
CTriangularMembershipFunction func3(8,8,10);
//--- Create wrappers for membership functions
double TriangularMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TriangularMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TriangularMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"TriangularMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TriangularMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TriangularMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(TriangularMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5]"
   graphic.CurveAdd(TriangularMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 10]
   graphic.CurveAdd(TriangularMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[8, 8, 10]
```

© 2000-2025, MetaQuotes Ltd.

```
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (Método Get)

Retorna el valor del primer punto en la abscisa.

```
double  X1()
```

Valor devuelto

Valor del primer punto en la abscisa.

X1 (Método Set)

Establece el valor del primer punto en la abscisa.

```
void  X1(
   const double  x1      // valor del primer punto en la abscisa
   )
```

Parámetros

x1

[in] Valor del primer punto en la abscisa.

X2 (Método Get)

Retorna valor del segundo punto en la abscisa.

```
double  X2()
```

Valor devuelto

Valor del segundo punto en la abscisa.

X2 (Método Set)

Establece el valor del segundo punto en la abscisa.

```
void  X2(
```

© 2000-2025, MetaQuotes Ltd.

```
   const double  x2      // valor del segundo punto en la abscisa
   )
```

Parámetros

x2

[in] Valor del segundo punto en la abscisa.

X3 (Método Get)

Retorna el valor del tercer punto en la abscisa.

```
double  X3()
```

Valor devuelto

Valor del tercer punto en la abscisa.

X3 (Método Set)

Establece el valor del tercer punto en la abscisa.

```
void  X3(
   const double  x3      // valor del tercer punto en la abscisa
   )
```

Parámetros

x3

[in] Valor del tercer punto en la abscisa.

ToNormalMF

Transforma la función triangular de pertenencia en una función de Gauss.

```
CNormalMembershipFunction*  ToNormalMF()
```

Valor devuelto

Puntero a la función de pertenencia de Gauss.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

© 2000-2025, MetaQuotes Ltd.

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

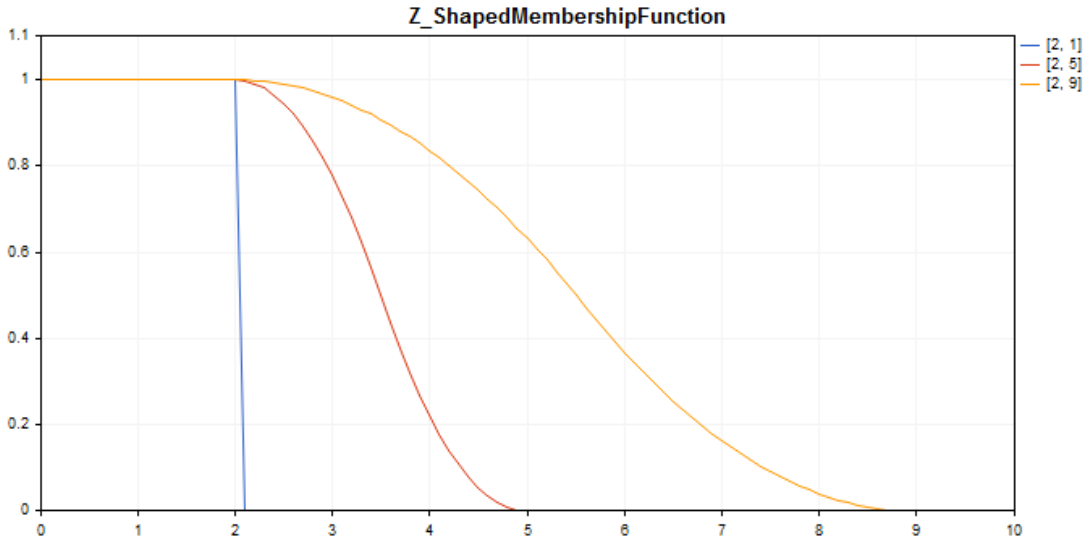
CZ_ShapedMembershipFunction

Clase para la implementación de una función de pertenencia en forma de Z con los parámetros A y B.

Descripción

La función crea una función de pertenencia en forma de Z de dos parámetros. Se trata de una función no creciente de pertenencia que adopta valores de 1 a 0. Los parámetros de la función definen un intervalo dentro del cual la función decrece de forma no lineal de 0 a 1.

Con la ayuda de la función se pueden presentar conjuntos difusos del tipo "muy bajo". Es decir, crea funciones no crecientes de pertenencia con saturación.



Un ejemplo de código para construir este gráfico se muestra más abajo.

Declaración

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject

IMembershipFunction

CZ_ShapedMembershipFunction

Métodos de clase

Método de clase	Descripción
A	Retorna y establece el parámetro del comienzo del intervalo de decrecimiento.

© 2000-2025, MetaQuotes Ltd.

Método de clase	Descripción
B	Retorna y establece el parámetro de la finalización del intervalo de decrecimiento.
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ejemplo

```
//+------------------------------------------------------------------+
//|                                  Z_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CZ_ShapedMembershipFunction func1(2,1);
CZ_ShapedMembershipFunction func2(2,5);
CZ_ShapedMembershipFunction func3(2,9);
//--- Create wrappers for membership functions
double Z_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double Z_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double Z_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"Z_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"Z_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("Z_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(Z_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(Z_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 5]");
   graphic.CurveAdd(Z_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 9]");
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Método Get)

Retorna el parámetro del comienzo del intervalo de decrecimiento.

```
double  A()
```

Valor devuelto

Parámetro del comienzo del intervalo de decrecimiento.

A (Método Set)

Establece el parámetro del comienzo del intervalo de decrecimiento.

```
void  A(
   const double  a      // parámetro del comienzo del intervalo de decrecimiento
   )
```

Parámetros

a

[in] Parámetro del comienzo del intervalo de decrecimiento.

B (Método Get)

Retorna el parámetro de la finalización del intervalo de decrecimiento.

```
double  B()
```

Valor devuelto

Parámetro de la finalización del intervalo de decrecimiento.

B (Método Set)

Establece el parámetro de la finalización del intervalo de decrecimiento.

```
void  B(
   const double  b      // parámetro de la finalización del intervalo de decrecimiento
   )
```

© 2000-2025, MetaQuotes Ltd.

Parámetros

b

[in] Parámetro de la finalización del intervalo de decrecimiento.

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

IMembershipFunction

Clase básica para todas las clases de las funciones de pertenencia.

Declaración

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Encabezamiento

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Jerarquía de herencia

CObject

IMembershipFunction

Descendientes directos

CCompositeMembershipFunction, CConstantMembershipFunction, CDifferencTwoSigmoidalMembershipFunction, CGeneralizedBellShapedMembershipFunction, CNormalCombinationMembershipFunction, CNormalMembershipFunction, CP_ShapedMembershipFunction, CProductTwoSigmoidalMembershipFunctions, CS_ShapedMembershipFunction, CSigmoidalMembershipFunction, CTrapezoidMembershipFunction, CTriangularMembershipFunction, CZ_ShapedMembershipFunction

Métodos de clase

Método de clase	Descripción
GetValue	Calcula el valor de la función de pertenenecia según el argumento indicado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

GetValue

Calcula el valor de la función de pertenenecia según el argumento indicado.

```
double  GetValue(
   const  x      // argumento de la función de pertenencia
   )
```

Parámetros

x

[in] Argumento de la función de pertenencia.

Valor devuelto

Valor de la función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

Reglas de los sistemas difusos

Se llama **sistema difuso** (sistema de deducción de lógica difusa) a la obtención de una conclusión en forma de conjunto difuso que corresponde a los valores actuales de las entradas usando un conjunto de **reglas difusas** y operaciones difusas.

Las **reglas difusas** definen la relación mutua entre las entradas y salidas del objeto investigado. El número de reglas en el sistema no está limitado. El formato generalizado de reglas difusas es el siguiente:

Si *se da la condición (envío) de la regla*, **entonces** *existe conclusión conforme a la regla*.

La *condición de la regla* caracteriza el estado actual del objeto. La *conclusión de la regla* caracteriza la forma en que la condición influye en el objeto.

Clase de normas para los sistemas difusos	Descripción
CMamdaniFuzzyRule	Clase para la implementación de la regla de lógica difusa para el algoritmo Mamdani
CSugenoFuzzyRule	Clase para la implementación de la regla de lógica difusa para el algoritmo Sugeno
CSingleCondition	La clase define la condición difusa expresada por la pareja "Variable difusa — Término difuso".
CConditions	La clase define el conjunto de condiciones difusas relacionadas con el operador.
CGenericFuzzyRule	Clase básica para implementar ambos tipos de reglas difusas.

© 2000-2025, MetaQuotes Ltd.

CMamdaniFuzzyRule

La deducción de lógica difusa del tipo Mamdani es uno de los dos tipos básicos de sistemas difusos. Los valores de la variable de salida en el mismo se indican con términos difusos.

Descripción

La regla lógica difusa para el algoritmo Mamdani se puede describir con la siguiente expresión:

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

donde:

- X = (X1, X2, X3 ... Xn) — vector de las variables de salida;
- Y — variable de salida;
- a = (a1, a2, a3 ... an) — vector de los valores de las variables de entrada;
- d — valor de la variable de salida;
- W — peso de la regla.

Declaración

```
class CMamdaniFuzzyRule : public CGenericFuzzyRule
```

Encabezamiento

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Jerarquía de herencia

CObject
IParsableRule
CGenericFuzzyRule
CMamdaniFuzzyRule

Métodos de clase

Método de clase	Descripción
Conclusion	Retorna y establece la conclusión de la regla difusa de Mamdani.
Weight	Retorna y establece el peso de la regla difusa de Mamdani.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

© 2000-2025, MetaQuotes Ltd.

Conclusion (Método Get)

Retorna la conclusión de la regla difusa de Mamdani.

```
CSingleConditon*  Conclusion()
```

Valor devuelto

Conclusión de la regla difusa.

Conclusion (Método Set)

Establece la conclusión de la regla difusa de Mamdani.

```
void  Conclusion(
   CSingleConditon*  value      // conclusión de la regla difusa de Mamdani
   )
```

Parámetros

value

[in] Conclusión de la regla difusa de Mamdani.

Weight (Метод Get)

Retorna el peso de la regla difusa de Mamdani.

```
double  Weight()
```

Valor devuelto

Peso de la regla difusa de Mamdani.

Weight (Método Set)

Establece el peso de la regla difusa de Mamdani.

```
void  Weight(
   const double  value      // peso de la regla difusa de Mamdani
   )
```

Parámetros

value

[in] Peso de la regla difusa de Mamdani.

© 2000-2025, MetaQuotes Ltd.

CSugenoFuzzyRule

La deducción de lógica difusa del tipo Sugeno es uno de los dos tipos básicos de sistemas difusos. Los valores de la variable de salida se establecen como una combinación lineal de parámetros de entrada.

Descripción

La diferencia con respecto a la regla de Mamdani reside en que el valor de la variable de entrada se establece, no con un término difuso, sino con la función lineal de la entrada. La regla lógica difusa para el algoritmo Sugeno se puede describir con la siguiente expresión:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

donde:

- X = (X1, X2, X3 ... Xn) — vector de las variables de salida;
- Y — variable de salida;
- a = (a1, a2, a3 ... an) — vector de los valores de las variables de entrada;
- b = (b1, b2, b3 ... bn) — coeficiente con un miembro libre en la función lineal para el valor de salida
- W — peso de la regla.

Declaración

```
class CSugenoFuzzyRule : public CGenericFuzzyRule
```

Encabezamiento

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Jerarquía de herencia

CObject
  IParsableRule
    CGenericFuzzyRule
      CSugenoFuzzyRule

Métodos de clase

Método de clase	Descripción
Conclusion	Retorna y establece la conclusión de la regla difusa de Sugeno.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

© 2000-2025, MetaQuotes Ltd.

Conclusion (Método Get)

Retorna la conclusión de la regla difusa de Sugeno.

```
CSingleConditon*  Conclusion()
```

Valor devuelto

Conclusión de la regla difusa de Sugeno.

Conclusion (Método Set)

Establece la conclusión de la regla difusa de Sugeno.

```
void  Conclusion(
   CSingleCondition*  value      // conclusión de la regla difusa de Sugeno
   )
```

Parámetros

value

[in] Conclusión de la regla difusa de Sugeno.

© 2000-2025, MetaQuotes Ltd.

CSingleCondition

La clase define la condición difusa expresada por la pareja "Variable difusa — Término difuso".

Descripción

Según la condición difusa, a una variable le corresponde un término. La condición difusa se puede describir con la siguiente expresión: *X is a*,

donde:

- X — variable difusa;
- a — valor de la variable difusa (término difuso).

Declaración

```
class CSingleCondition : public ICondition
```

Encabezamiento

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Jerarquía de herencia

CObject
  ICondition
    CSingleCondition

Descendientes directos

CFuzzyCondition

Métodos de clase

Método de clase	Descripción
Not	Retorna y establece la bandera que indica si hay que aplicar la negación a esta condición.
Term	Retorna y establece un término difuso para esta condición.
Var	Retorna y establece una variable difusa para esta condición.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Not (Método Get)

Retorna la bandera que indica si hay que aplicar la negación a esta condición.

```
bool Not()
```

© 2000-2025, MetaQuotes Ltd.

Valor devuelto

Valor de la bandera.

Not (Método Set)

Establece la bandera que indica si hay que aplicar la negación a esta condición.

```
void  Not(
   bool  not      // valor de la bandera
   )
```

Parámetros

not

[in] Valor de la bandera.

Term (Método Get)

Retorna un término difuso para esta condición.

```
INamedValue* Term()
```

Valor devuelto

Término difuso para esta condición.

Term (Método Set)

Establece un término difuso para esta condición.

```
void  Term(
   INamedValue*&  value      // Término difuso para esta condición
   )
```

Parámetros

value

[in] Término difuso para esta condición.

Var (Método Get)

Retorna una variable difusa para esta condición.

```
INamedVariable* Var()
```

Valor devuelto

Variable difusa para esta condición.

Var (Método Set)

Establece una variable difusa para esta condición.

© 2000-2025, MetaQuotes Ltd.

```
void  Var(
   INamedVariable*&  value      // variable difusa para esta condición
   )
```

Parámetros

value

[in] variable difusa.

© 2000-2025, MetaQuotes Ltd.

CConditions

La clase define el conjunto de condiciones difusas relacionadas con el operador.

Descripción

El conjunto de condiciones difusas relacionadas con el operador puede ser descrito, por ejemplo, con la siguiente expresión:

$$(X_1 \; is \; a_1) \wedge (X_2 \; is \; a_2) \wedge \ldots \wedge (X_n \; is \; a_n)$$

donde:

- X = (X1, X2, X3 ... Xn) — vector de las variables de entrada;
- a = (a1, a2, a3 ... an) — vector de los valores de las variables de entrada.

En este ejemplo se ha usado el operador *and* (i). Asimismo, en esta clase está disponible el operador *or* (o).

Declaración

```
class CConditions : public ICondition
```

Encabezamiento

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Jerarquía de herencia

CObject
  ICondition
    CConditions

Métodos de clase

Método de clase	Descripción
ConditionsList	Retorna la lista de todas las condiciones
Not	Retorna y establece la bandera que indica si hay que aplicar la negación a estas condiciones
Op	Retorna y establece el tipo de operador de implicación de condiciones.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

ConditionsList

Retorna la lista de todas las condiciones.

© 2000-2025, MetaQuotes Ltd.

```
CList*  ConditionsList()
```

Valor devuelto

Lista de todas las condiciones.

Not (Método Get)

Retorna la bandera que indica si hay que aplicar la negación a estas condiciones

```
bool Not()
```

Valor devuelto

Valor de la bandera.

Not (Método Set)

Establece la bandera que indica si hay que aplicar la negación a estas condiciones

```
void  Not(
   bool  not      // valor de la bandera
   )
```

Parámetros

not

[in] Valor de la bandera.

Op (Método Get)

Retorna el tipo de operador de implicación de condiciones. Están disponibles dos operadores: *and* (y) y *or* (o).

```
OperatorType Op()
```

Valor devuelto

Tipo de operador de implicación de condiciones.

Op (Método Set)

Establece el operador de implicación de condiciones. Están disponibles dos operadores: *and* (y) y *or* (o).

```
void  Op(
   OperatorType  op     // Tipo de operador de implicación de condiciones
   )
```

Parámetros

op

© 2000-2025, MetaQuotes Ltd.

[in] Tipo de operador de implicación de condiciones.

© 2000-2025, MetaQuotes Ltd.

CGenericFuzzyRule

Clase básica para ambos tipos de reglas difusas.

Declaración

```
class CGenericFuzzyRule : public IParsableRule
```

Encabezamiento

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Jerarquía de herencia

CObject

IParsableRule

CGenericFuzzyRule

Descendientes directos

CMamdaniFuzzyRule, CSugenoFuzzyRule

Métodos de clase

Método de clase	Descripción
Conclusion	Retorna y establece la conclusión de la regla difusa.
Condition	Retorna y establece la condición (conjunto de condiciones) 'if' para la regla difusa
CreateCondition	Crea la condición para la regla difusa según los parámetros establecidos

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Conclusion (Método Get)

Retorna la conclusión de la regla difusa.

```
CSingleConditon*  Conclusion()
```

Valor devuelto

Conclusión de la regla difusa.

Conclusion (Método Set)

Establece la conclusión de la regla difusa.

```
virtual void  Conclusion(
   CSingleConditon*  value      // conclusión de la regla difusa
```

© 2000-2025, MetaQuotes Ltd.

```
   )
```

Parámetros

value

[in] Conclusión de la regla difusa.

Condition (Método Get)

Retorna la condición (conjunto de condiciones) 'if' para la regla difusa.

```
CConditons*  Condition()
```

Valor devuelto

Regla difusa (conjunto de condiciones).

Condition (Método Set)

Establece la condición (conjunto de condiciones) 'if' para la regla difusa

```
void  Condition(
   CConditons*  value      // condición (conjunto de condiciones) 'if' para la regla d
   )
```

Parámetros

value

[in] Condición difusa (conjunto de condiciones).

CreateCondition

Crea la condición para la regla difusa según los parámetros establecidos.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,       // variable difusa
   CFuzzyTerm*      term,      // término difuso
   )
```

Parámetros

var

[in] Variable difusa.

term

[in] Término difuso.

Valor devuelto

Estado de la regla difusa.

CreateCondition

© 2000-2025, MetaQuotes Ltd.

Crea la condición para la regla difusa según los parámetros establecidos.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,      // variable difusa
   CFuzzyTerm*      term,     // término difuso
   bool             not,      // bandera que indica si hay que aplicar la negación a l
   )
```

Parámetros

var

[in] Variable difusa.

term

[in] Término difuso.

not

[in] bandera que indica si hay que aplicar la negación a la condición.

Valor devuelto

Estado de la regla difusa.

CreateCondition

Crea la condición para la regla difusa según los parámetros establecidos.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable* var,      // variable difusa
   CFuzzyTerm*     term,     // término difuso
   bool            not,      // bandera que indica si hay que aplicar la negación a la
   HedgeType       hedge     // tipo de implicación de la condición
   )
```

Parámetros

var

[in] Variable difusa.

term

[in] Término difuso.

not

[in] bandera que indica si hay que aplicar la negación a la condición.

hedge

[in] Tipo de implicación de la condición.

Valor devuelto

Estado de la regla difusa.

© 2000-2025, MetaQuotes Ltd.

Variables para los sistemas difusos

En los sistemas difusos se usan **variables difusas (lingüísticas)**. Se trata de aquellas variables cuyos valores pueden ser palabras o combinaciones de palabras de un cierto lenguaje natural o artificial.

Las variables lingüísticas constituyen los conjuntos difusos. A la hora de definir conjuntos difusos, el carácter y el número de las variables difusas cambian para cada tarea por separado.

Clase	Descripción
CFuzzyVariable	Clase con cuya ayuda se crean las variables difusas de tipo general.
CSugenoVariable	Clase con cuya ayuda se crean las variables difusas de tipo Sugeno.

© 2000-2025, MetaQuotes Ltd.

CFuzzyVariable

Clase con cuya ayuda se crean las variables difusas de tipo general.

Descripción

Una variable difusa, en nuestro caso, se define con los siguientes parámetros:

- valor máximo de la variable.
- valor mínimo de la variable;
- nombre de la variable difusa;
- conjunto de términos (conjunto de todos los valores posibles que puede adoptar una variable lingüística).

Declaración

```
class CFuzzyVariable : public CNamedVariableImpl
```

Encabezamiento

```
#include <Math\Fuzzy\fuzzyvariable.mqh>
```

Jerarquía de herencia

CObject
INamedValue
INamedVariable
CNamedVariableImpl
CFuzzyVariable

Métodos de clase

Método de clase	Descripción
AddTerm	Añade un término difuso a la variable difusa.
GetTermByName	Obtiene un término difuso según el nombre establecido.
Max	Retorna y establece el valor máximo para una variable difusa.
Min	Retorna y establece el valor mínimo para una variable difusa.
Terms	Retorna y establece la lista de términos difusos para una variable difusa dada.
Values	Retorna y establece la lista de términos difusos para una variable difusa dada.

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CNamedVariableImpl

Name, Name

AddTerm

Añade un término difuso a una variable difusa.

```
void  AddTerm(
   CFuzzyTerm*&  term      // término difuso
   )
```

Parámetros

term

[in] Término difuso.

GetTermByName

Obtiene un término difuso según el nombre establecido.

```
CFuzzyTerm*  GetTermByName(
   const string  name      // nombre del término difuso
   )
```

Parámetros

name

[in] Nombre del término difuso.

Valor devuelto

Término difuso con el nombre establecido.

Max (Método Get)

Retorna el valor máximo para una variable difusa.

```
double  Max()
```

Valor devuelto

Valor máximo para una variable difusa.

Max (Método Set)

Establece el valor máximo para una variable difusa.

```
void  Max(
   const double  max      // valor máximo para una variable difusa
```

© 2000-2025, MetaQuotes Ltd.

```
   )
```

Parámetros

max

[in] Valor máximo para una variable difusa.

Min (Método Get)

Retorna el valor mínimo para una variable difusa.

```
double  Min()
```

Valor devuelto

Valor mínimo para una variable difusa.

Max (Método Set)

Establece el valor mínimo para una variable difusa.

```
void  Min(
   const double  min      // valor mínimo para una variable difusa
   )
```

Parámetros

min

[in] Valor mínimo para una variable difusa.

Terms (Método Get)

Retorna la lista de términos difusos para una variable difusa dada.

```
CList*  Terms()
```

Valor devuelto

Lista de términos difusos para una variable difusa dada.

Terms (Método Set)

Establece la lista de términos difusos para una variable difusa dada.

```
void  Terms(
   CList*&  terms      // lista de términos difusos para una variable difusa dada
   )
```

Parámetros

terms

[in] lista de términos difusos para una variable difusa dada.

© 2000-2025, MetaQuotes Ltd.

Values

Retorna la lista de términos difusos para una variable difusa dada.

```
CList*  Values()
```

Valor devuelto

Lista de términos difusos para una variable dada.

© 2000-2025, MetaQuotes Ltd.

CSugenoVariable

Clase con cuya ayuda se crean las variables difusas de tipo Sugeno.

Descripción

La variable difusa del tipo Sugeno se distingue de la variable lingüística de tipo general en que se establece, no un conjunto de términos, sino un conjunto de funciones lineales.

Declaración

```
class CSugenoVariable : public CNamedVariableImpl
```

Encabezamiento

```
#include <Math\Fuzzy\sugenovariable.mqh>
```

Jerarquía de herencia

CObject
INamedValue
INamedVariable
CNamedVariableImpl
CSugenoVariable

Métodos de clase

Método de clase	Descripción
Functions	Retorna la lista de funciones lineales de la variable difusa de Sugeno.
GetFuncByName	Retorna una función lineal según el nombre dado.
Values	Retorna la lista de funciones lineales de la variable difusa de Sugeno.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CNamedVariableImpl

Name, Name

Functions

Retorna la lista de funciones lineales de la variable difusa de Sugeno.

```
CList*  Functions()
```

Valor devuelto

© 2000-2025, MetaQuotes Ltd.

Lista de funciones lineales.

GetFuncByName

Retorna una función lineal según el nombre dado.

```
ISugenoFunction*  GetFuncByName(
   const string  name      // nombre de la función lineal
   )
```

Parámetros

name

[in] Nombre de la función lineal.

Valor devuelto

Función lineal con el nombre establecido.

Values

Retorna la lista de funciones lineales de la variable difusa de Sugeno.

```
CList*  Values()
```

Valor devuelto

Lista de funciones lineales de la variabe difusa de Sugeno.

© 2000-2025, MetaQuotes Ltd.

CFuzzyTerm (Términos difusos)

Clase para la implementación de términos difusos.

Descripción

Se llama término (term) a cualquier elemento del conjunto de términos. Un término se define por dos componentes:

- el nombre del término difuso;
- la función de pertenencia.

Declaración

```
class CFuzzyTerm : public CNamedValueImpl
```

Encabezamiento

```
#include <Math\Fuzzy\fuzzyterm.mqh>
```

Jerarquía de herencia

CObject
  INamedValue
    CNamedValueImpl
      CFuzzyTerm

Métodos de clase

Método de clase	Descripción
MembershipFunction	Retorna la función de pertenencia para el término difuso dado.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CNamedValueImpl

Name, Name

© 2000-2025, MetaQuotes Ltd.

MembershipFunction

Retorna la función de pertenencia para el término difuso dado.

```
IMembershipFunction*  MembershipFunction()
```

Valor devuelto

Función de pertenencia.

© 2000-2025, MetaQuotes Ltd.

Sistemas difusos

Un **sistema difuso** (o *modelo difuso*) es un modelo matemático en cuya base de cálculo se encuentra la lógica difusa. Se suele recurrir a este tipo de construcciones cuando el objeto de investigación tiene una formalización muy débil, y su descripción matemática exacta es demasiado compleja o desconocida.

El orden de construcción del modelo se puede dividir en tres etapas:

1. Definición de los parámetros de entrada y salida del modelo.
2. Construcción de la base de conocimiento del modelo.
3. Elección de uno de los métodos de deducción de lógica difusa (Mamdani o Sugeno).

Es precisamente de la primera etapa de lo que dependen las otras dos, pues define el futuro funcionamiento del modelo.

La **base de conocimientos** (*base de reglas*) es un conjunto de normas del tipo: "si, entonces", que definen la relación mutua entre las entradas y salidas del objeto investigado.

La condición (Condition) de la regla caracteriza el estado actual del objeto y la **conclusión (Conclusion)** es la forma en que esta condición influye en el objeto.

Las condiciones y conclusiones de cada regla pueden ser de dos tipos:

1. simple (referencia a Csinglcond), cuando en ella participa una variable difusa;
2. compuesto (referencia a Cconditions), cuando participan varias variables difusas.

Cada regla en el sistema tiene su **peso**, es decir, el valor de la regla en el modelo. Los coeficientes de peso se asignan a la regla en el rango [0, 1].

Dependiendo de la base de conocimiento creada para el modelo, se definirá un sistema de dedución de lógica difusa. Se llama **dedución de lógica difusa** a la obtención de una conclusión en forma de conjunto difuso que corresponde a los valores actuales de las entradas usando una base difusa de conocimiento y operaciones difusas. Existen dos tipos principales de dedución de lógica difusa: Mamdani y Sugeno.

© 2000-2025, MetaQuotes Ltd.

Sistema Mamdani

Los valores de la variable de salida en el sistema Mamdani se indican con términos difusos.

Descripción

La regla lógica difusa para el algoritmo Mamdani se puede describir con la siguiente expresión:

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

donde:

- X = (X1, X2, X3 ... Xn) — vector de las variables de salida;
- Y — variable de salida;
- a = (a1, a2, a3 ... an) — vector de los valores de las variables de entrada;
- d — valor de la variable de salida;
- W — peso de la regla.

Métodos de clase

Método de clase	Descripción
AggregationMethod	Establece el tipo de agregación de las condiciones.
Calculate	Calcula la deducción de lógica difusa para este sistema
DefuzzificationMethod	Establece el tipo del método de defusificación
EmptyRule	Crea una regla difusa vacía de Mamdani basada en el sistema actual
ImplicationMethod	Establece el tipo de operador de implicación del sistema
Output	Retorna la lista de variables difusas de salida Mamdani.
OutputByName	Retorna una variable difusa de salida Mamdani según el nombre indicado.
ParseRule	Crea una regla difusa de Mamdani basada en una línea establecida.
Rules	Retorna la lista de reglas difusas de Mamdani.

Métodos heredados de la clase CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

AggregationMethod

Establece el tipo de método de agregación de condiciones.

```
void  AggregationMethod(
```

© 2000-2025, MetaQuotes Ltd.

```
   AggregationMethod  value      // tipo de método de agregación
   )
```

Parámetros

value

[in] Tipo de método de agregación de condiciones.

Calculate

Calcula la deducción de lógica difusa para este sistema.

```
CList*  Calculate(
   CList*  inputValues      // datos de entrada
   )
```

Parámetros

inputValues

[in] Datos de entrada para el cálculo.

Valor devuelto

Resultado de los cálculos.

DefuzzificationMethod

Establece el tipo de método de defusificación.

```
void  DefuzzificationMethod(
   DefuzzificationMethod  value      // tipo de método de defusificación.
   )
```

Parámetros

value

[in] Tipo de método de defusificación.

EmptyRule

Crea una regla difusa vacía de Mamdani basada en el sistema actual.

```
CMamdaniFuzzyRule*  EmptyRule()
```

Valor devuelto

Regla difusa de Mamdani.

ImplicationMethod

Establece el tipo de operador de implicación del sistema.

```
void  ImplicationMethod(
```

© 2000-2025, MetaQuotes Ltd.

```
   ImplicationMethod  value      // tipo de operador de implicación
   )
```

Parámetros

value

[in] Tipo de operador de la implicación de las condiciones.

Output

Retorna la lista de variables difusas de salida de Mamdani.

```
CList*  Output()
```

Valor devuelto

Lista de variables difusas.

OutputByName

Retorna una variable difusa de salida Mamdani según el nombre indicado.

```
CFuzzyVariable*  OutputByName(
   const string  name      // nombre de la variable difusa
   )
```

Parámetros

name

[in] Nombre de la variable difusa.

Valor devuelto

Variable difusa de Mamdani con el nombre establecido.

ParseRule

Crea una regla difusa de Mamdani basada en una línea establecida.

```
CMamdaniFuzzyRule*  ParseRule(
   const string  rule      // representación de línea de la regla difusa
   )
```

Parámetros

rule

[in] Representación de línea de la regla difusa.

Valor devuelto

Regla difusa de Mamdani.

© 2000-2025, MetaQuotes Ltd.

Rules

Retorna la lista de reglas difusas de Mamdani.

```
CList*  Rules()
```

Valor devuelto

Lista de reglas difusas de Mamdani.

© 2000-2025, MetaQuotes Ltd.

Sistema Sugeno

El sistema de lógica difusa del tipo Sugeno es uno de los dos tipos básicos de sistemas difusos. Los valores de la variable de salida se establecen como una combinación lineal de parámetros de entrada.

Descripción

La diferencia del sistema de Mamdani reside en que el valor de la variable de entrada se establece, no con un término difuso, sino con la función lineal de las entradas. La regla lógica difusa para el algoritmo Sugeno se puede describir con la siguiente expresión:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

donde:

- X = (X1, X2, X3 ... Xn) — vector de las variables de salida;
- Y — variable de salida;
- a = (a1, a2, a3 ... an) — vector de los valores de las variables de entrada;
- b = (b1, b2, b3 ... bn) — coeficiente con un miembro libre en la función lineal para el valor de salida
- W — peso de la regla.

Métodos de clase

Método de clase	Descripción
Calculate	Calcula la deducción de lógica difusa para este sistema.
CreateSugenoFunction	Crea una función lineal de Sugeno para el sistema dado.
EmptyRule	Crea una regla difusa vacía de Sugeno basada en el sistema actual.
Output	Retorna la lista de variables difusas de salida de Sugeno.
OutputByName	Retorna una variable difusa de salida de Sugeno según el nombre indicado.
ParseRule	Crea una regla difusa de Sugeno basada en una línea establecida.
Rules	Retorna la lista de reglas difusas.

Métodos heredados de la clase CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

Calculate

Calcula la deducción de lógica difusa para este sistema.

© 2000-2025, MetaQuotes Ltd.

```
CList*  Calculate(
   CList*&  inputValues      // datos de entrada
   )
```

Parámetros

inputValues

[in] Datos de entrada para el cálculo.

Valor devuelto

Resultado de los cálculos.

CreateSugenoFunction

Crea una función lineal de Sugeno para el sistema dado.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,          // nombre de la función
   const double& coeffs[]       // coeficientes de la función
   )
```

Parámetros

name

[in] Nombre de la función.

coeffs[]

[in] Coeficientes de la función.

Valor devuelto

Función lineal de Sugeno.

Nota

El tamaño de la matriz de los coeficientes puede ser o bien igual al número de parámetros de entrada, o bien superior al mismo en una unidad. En el primer caso, el miembro libre de la función lineal se Sugeno es igual a cero, y en el segundo, al último coeficiente.

CreateSugenoFunction

Crea una función lineal de Sugeno para el sistema dado.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,          // nombre de la función
   CList*&       coeffs,        // la lista de parejas de variable difusa es el coefic
   const double  constValue     // coeficiente
   )
```

Parámetros

name

[in] Nombre de la función.

© 2000-2025, MetaQuotes Ltd.

coeffs[]

[in] Coeficientes de las funciones.

Valor devuelto

Función lineal de Sugeno.

EmptyRule

Crea una regla difusa vacía de Sugeno basada en el sistema actual.

```
CSugenoFuzzyRule*  EmptyRule()
```

Valor devuelto

Regla difusa de Sugeno.

Output

Retorna la lista de variables difusas de salida de Sugeno.

```
CList*  Output()
```

Valor devuelto

Lista de variables difusas.

OutputByName

Retorna una variable difusa de salida de Sugeno según el nombre indicado.

```
CSugenoVariable*  OutputByName(
   const string  name      // nombre de la variable difusa
   )
```

Parámetros

name

Nombre de la variable difusa.

Valor devuelto

Variable difusa de Sugeno con el nombre establecido.

ParseRule

Crea una regla difusa de Sugeno basada en una línea establecida.

```
CSugenoFuzzyRule*  ParseRule(
   const string  rule      // representación de línea de la regla difusa de Sugeno
   )
```

Parámetros

© 2000-2025, MetaQuotes Ltd.

rule

[in] Representación de línea de la regla difusa de Sugeno.

Valor devuelto

Regla difusa de Sugeno.

Rules

Retorna la lista de reglas difusas.

```
CList*  Rules()
```

Valor devuelto

Lista de reglas difusas.

© 2000-2025, MetaQuotes Ltd.

Clase para trabajar con programas OpenCL

La clase COpenCL es un envoltorio para trabajar cómodamente con las funciones OpenCL. El uso de GPU en ciertos casos permite aumentar considerablemente la velocidad de cálculo.

Los ejemplos de uso de la clase para los cálculos en las cifras float y double se pueden encontrar en la carpeta MQL5\Scripts\Examples\OpenCL\, en los catálogos correspondientes, los códigos fuente de los propios programas OpenCL se encuentran en los subcatálogos MQL5\Scripts\Examples\OpenCL\Double\Kernels y MQL5\Scripts\Examples\OpenCL\Float\Kernels.

- MatrixMult.mq5 - ejemplo de multiplicación de matrices con uso de la memoria local y global
- BitonicSort.mq5 - ejemplo de clasificación paralela de los elementos de la matriz en GPU
- FFT.mq5 - ejemplo de cálculo de la transformada rápida de Fourier
- Wavelet.mq5 - ejemplo de cálculo de la transformada de ondícula de los datos con la ayuda de la ondícula de Morlet.

Es recomendable escribir el código fuente de OpenCL en archivos CL aparte, que después se puedan incluir en el programa MQL5 con la ayuda de las variables de recurso.

Declaración

```
class COpenCL
```

Encabezamiento

```
#include <OpenCL\COpenCL.mqh>
```

Métodos de la clase

Nombre	Descripción
BufferCreate	Crea un búfer OpenCL según el índice indicado
BufferFree	Elimina un búfer según el índice indicado
BufferFromArray	Crea un búfer según el índice indicado de la matriz de valores
BufferRead	Lee en la matriz un búfer OpenCL según el índice indicado
BufferWrite	Escribe la matriz de datos en el búfer según el índice indicado
Execute	Ejecuta un programa OpenCL según el índice indicado
GetContext	Retorna el manejador del contexto OpenCL
GetKernel	Retorna el manejador del objeto núcleo según el índice indicado
GetKernelName	Retorna el nombre del objeto núcleo según el índice indicado
GetProgram	Retorna el manejador de un programa OpenCL
Initialize	Inicia un programa OpenCL
KernelCreate	Crea un punto de entrada en un programa OpenCL según el índice indicado

© 2000-2025, MetaQuotes Ltd.

Nombre	Descripción
KernelFree	Elimina la función de inicio de OpenCL según el índice indicado.
SetArgument	Establece un parámetro para una función OpenCL según el índice indicado.
SetArgumentBuffer	Establece un búfer de OpenCL como parámetro de una función OpenCL según el índice indicado
SetArgumentLocalMemory	Establece un parámetro en la memoria local para una función OpenCL según el índice indicado.
SetBuffersCount	Establece el número de búferes
SetKernelsCount	Establece el número de objetos núcleo
Shutdown	Libera de la memoria un programa OpenCL.
SupportDouble	Aclara si se da soporte a los tipos reales de datos en el dispositivo

© 2000-2025, MetaQuotes Ltd.

BufferCreate

Crea un búfer OpenCL según el índice indicado.

```
bool  BufferCreate(
   const int   buffer_index,      // índice del búfer
   const uint  size_in_bytes,     // tamaño del búfer en bytes
   const uint  flags              // combinación de banderas que establecen las propie
   );
```

Parámetros

buffer_index

[in] Índice del búfer.

size_in_bytes

[in] Tamaño del búfer en bytes.

flags

[in] Propiedades del búfer establecidas con una combinación de banderas.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

BufferFree

Elimina un búfer según el índice indicado.

```
bool  BufferFree(
   const int  buffer_index      // índice del búfer
   );
```

Parámetros

buffer_index

[in] Índice del búfer.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

BufferFromArray

Crea un búfer según el índice indicado de la matriz de valores.

```
template<typename T>
bool  BufferFromArray(
   const int   buffer_index,          // índice del búfer
   T           &data[],               // matriz de valores
   const uint  data_array_offset,     // desplazamiento en la matriz de valores en byt
   const uint  data_array_count,      // número de valores de la matriz para la escrit
   const uint  flags                  // combinación de banderas que establecen las pr
   );
```

Parámetros

buffer_index

[in] Índice del búfer.

&data[]

[in] Matriz de valores que hay que escribir en el búfer OpenCL.

data_array_offset

[in] Desplazamiento en la matriz de valores en bytes, desde donde comienza la escritura.

data_array_count

[in] Número de valores a escribir.

flags

[in] Propiedades del búfer establecidas con una combinación de banderas.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

BufferRead

Lee en la matriz un búfer OpenCL según el índice indicado.

```
template<typename T>
bool  BufferRead(
   const int   buffer_index,          // índice del búfer
   T           &data[],               // matriz de valores
   const uint  cl_buffer_offset,      // desplazamiento en el búfer OpenCL en bytes
   const uint  data_array_offset,     // desplazamiento en la matriz en elementos
   const uint  data_array_count       // número de valores del búfer para la lectura
   );
```

Parámetros

buffer_index

[in] Índice del búfer.

&data[]

[in] Matriz para obtener los valores del búfer OpenCL.

cl_buffer_offset

[in] Desplazamiento en el búfer OpenCL en bytes, desde donde comienza la lectura.

data_array_offset

[in] Índice del primer elemento de la matriz para la escritura de los valores del búfer OpenCL.

data_array_count

[in] Número de valores a escribir.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

BufferWrite

Escribe la matriz de datos en el búfer según el índice indicado.

```
template<typename T>
bool  BufferWrite(
   const int   buffer_index,          // índice del búfer
   T           &data[],               // matriz de valores
   const uint  cl_buffer_offset,      // desplazamiento en el búfer OpenCL en bytes
   const uint  data_array_offset,     // desplazamiento en la matriz en elementos
   const uint  data_array_count       // número de valores de la matriz para la escrit
   );
```

Parámetros

buffer_index

[in] Índice del búfer.

&data[]

[in] Matriz de valores que hay que escribir en el búfer OpenCL.

cl_buffer_offset

[in] Desplazamiento en el búfer OpenCL en bytes, desde donde comienza la escritura.

data_array_offset

[in] Índice del primer elemento de la matriz, a partir del cual se toman los valores de la matriz para la escritura en el búfer OpenCL.

data_array_count

[in] Número de valores a escribir.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

Execute

Ejecuta un programa OpenCL según el índice indicado.

```
bool  Execute(
   const int   kernel_index,           // índice del núcleo
   const int   work_dim,               // dimensión del espacio de tareas
   const uint  &work_offset[],         // desplazamiento inicial en el espacio de tare
   const uint  &work_size[]            // número total de tareas
   );
```

Ejecuta un programa OpenCL según el índice indicado con el número de tareas establecido en el grupo local.

```
bool  Execute(
   const int   kernel_index,           // índice del núcleo
   const int   work_dim,               // dimensión del espacio de tareas
   const uint  &work_offset[],         // desplazamiento inicial en el espacio de tare
   const uint  &work_size[],           // número total de tareas
   const uint  &local_work_size[]      // número de tareas en el grupo local
   );
```

Parámetros

kernel_index

[in] Índice del objeto núcleo.

work_dim

[in] Dimensión del espacio de tareas.

&work_offset[]

[in] Desplazamiento inicial en el espacio de tareas.

&work_size[]

[in] Tamaño del subconjunto de tareas.

&local_work_size[]

[in] Tamaño del subconjunto local de tareas en el grupo.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

GetContext

Retorna el manejador del contexto OpenCL.

```
int  GetContext();
```

Valor devuelto

Manejador del contexto OpenCL.

© 2000-2025, MetaQuotes Ltd.

GetKernel

Retorna el manejador del objeto núcleo según el índice indicado.

```
int  GetKernel(
   const int  kernel_index      // índice del núcleo
   );
```

Parámetros

kernel_index

[in] Índice del objeto núcleo.

Valor devuelto

Manejador del objeto núcleo.

© 2000-2025, MetaQuotes Ltd.

GetKernelName

Retorna el nombre del objeto núcleo según el índice indicado.

```
string  GetKernelName(
   const int  kernel_index      // índice del núcleo
   );
```

Parámetros

kernel_index

[in] Índice del objeto núcleo.

Valor devuelto

Nombre del objeto núcleo.

© 2000-2025, MetaQuotes Ltd.

GetProgram

Retorna el manejador de un programa OpenCL.

```
int  GetProgram();
```

Valor devuelto

Manejador del programa OpenCL.

© 2000-2025, MetaQuotes Ltd.

Initialize

Inicializa un programa OpenCL.

```
bool  Initialize(
   const string  program,           // manejador del programa OpenCL
   const bool    show_log=true      // llevar registro del log
   );
```

Parámetros

program

[in] Manejador del programa OpenCL.

show_log=true

[in] Activar el registro de los mensajes en el diario.

Valor devuelto

Retorna true, si la inicialización ha tenido éxito. En caso contrario, retorna false.

© 2000-2025, MetaQuotes Ltd.

KernelCreate

Crea un punto de entrada en un programa OpenCL según el índice indicado.

```
bool  KernelCreate(
   const int     kernel_index,     // índice del núcleo
   const string  kernel_name       // nombre del núcleo
   );
```

Parámetros

kernel_index

[in] Índice del objeto núcleo.

kernel_name

[in] Nombre del objeto núcleo.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

KernelFree

Elimina la función de inicio de OpenCL según el índice indicado.

```
bool  KernelFree(
   const int  kernel_index      // índice del núcleo
   );
```

Parámetros

kernel_index

[in] Índice del objeto núcleo.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SetArgument

Establece un parámetro para una función OpenCL según el índice indicado.

```
template<typename T>
bool  SetArgument(
   const int  kernel_index,     // índice del núcleo
   const int  arg_index,        // índice del argumento de la función
   T          value             // código fuente
   );
```

Parámetros

kernel_index

[in] Índice del objeto núcleo.

arg_index

[in] Índice del argumento de la función.

value

[in] Valor del argumento de la función.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SetArgumentBuffer

Establece un búfer de OpenCL como parámetro de una función OpenCL según el índice indicado

```
bool  SetArgumentBuffer(
   const int  kernel_index,     // índice del núcleo
   const int  arg_index,        // índice del argumento de la función
   const int  buffer_index      // índice del búfer
   );
```

Parámetros

kernel_index

[in] Índice del objeto núcleo.

arg_index

[in] Índice del argumento de la función.

buffer_index

[in] Índice del búfer.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SetArgumentLocalMemory

Establece un parámetro en la memoria local para una función OpenCL según el índice indicado.

```
bool  SetArgumentLocalMemory(
   const int  kernel_index,          // índice del núcleo
   const int  arg_index,             // índice del argumento de la función
   const int  local_memory_size      // tamaño de la memoria local
   );
```

Parámetros

kernel_index

[in] Índice del objeto núcleo.

arg_index

[in] Índice del argumento de la función.

local_memory_size

[in] Tamaño de la memoria local.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SetBuffersCount

Establece el número de búferes.

```
bool  SetBuffersCount(
   const int  total_buffers      // número de búferes
   );
```

Parámetros

total_buffers

[in] número total de búferes.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SetKernelsCount

Establece el número de objetos núcleo.

```
bool  SetKernelsCount(
   const int  total_kernels      // número de núcleos
   );
```

Parámetros

total_kernels

[in] Número total de núcleos.

Valor devuelto

En caso de que la ejecución tenga éxito, devuelve true, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

Shutdown

Libera de la memoria un programa OpenCL.

```
void  Shutdown();
```

Valor devuelto

No hay valores devueltos.

© 2000-2025, MetaQuotes Ltd.

SupportDouble

Aclara si se da soporte a los tipos reales de datos en el dispositivo.

```
bool  SupportDouble();
```

Valor devuelto

Retorna true, si en el dispositivo se da soporte a los tipos reales de datos.

© 2000-2025, MetaQuotes Ltd.

Clase CObject básica

La clase CObject es la clase base sobre la que se construye la Librería Estándar de MQL5.

Descripción

La clase CObject permite que sus descendientes formen parte de una lista enlazada. También identifica una serie de métodos virtuales para la posterior ejecución en clases descendientes.

Declaración

```
class CObject
```

Título

```
#include <Object.mqh>
```

Jerarquía de herencia

CObject

Descendientes directos

CAccountInfo, CArray, CChart, CChartObject, CCurve, CDealInfo, CDictionary_Obj_Double, CDictionary_Obj_Obj, CDictionary_String_Obj, CExpertBase, CFile, CHistoryOrderInfo, CList, COrderInfo, CPositionInfo, CString, CSymbolInfo, CTerminalInfo, CTrade, CTreeNode, CWnd, ICondition, IExpression, IMembershipFunction, INamedValue, IParsableRule

Métodos de la clase

Atributos	
Prev	Obtiene el valor del ítem anterior
Prev	Establece el valor del ítem anterior
Next	Obtiene el valor del elemento posterior
Next	Establece el elemento siguiente
Métodos de comparación	
virtual Compare	Devuelve el resultado de la comparación con otro objeto
Entrada/salida	
virtual Save	Escribe el objeto en el archivo
virtual Load	Lee el objeto del archivo
virtual Type	Devuelve el tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Prev

Obtiene un puntero al ítem anterior de la lista.

```
CObject*  Prev()
```

Valor devuelto

Puntero al ítem anterior de la lista. Si el ítem sale listado el primero, entonces devuelve NULL.

Ejemplo:

```
//--- ejemplo de CObject::Prev()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Error en la creación del objeto");
      delete object_first;
      return;
     }
   //--- establece la interconexión
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utiliza el objeto anterior
   CObject *object=object_second.Prev();
   //--- borra los objetos
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Prev

Coloca el puntero en el elemento anterior de la lista.

```
void  Prev(
   CObject*  object      // Puntero al elemento anterior de la lista
   )
```

Parámetros

object

[in] Nuevo valor del puntero al elemento anterior de la lista.

Ejemplo:

```
//--- ejemplo de CObject::Prev(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Error en la creación del objeto");
      delete object_first;
      return;
     }
   //--- establece la interconexión
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utiliza los objetos
   //--- ...
   //--- borra los objetos
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Next

Obtiene un puntero al siguiente elemento de la lista.

```
CObject*  Next()
```

Valor devuelto

Puntero al siguiente elemento de la lista. Si es el último elemento de la lista, devuelve NULL.

Ejemplo:

```
//--- ejemplo de CObject::Next()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Error en la creación del objeto");
      delete object_first;
      return;
     }
   //--- establece la interconexión
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utiliza el siguiente objeto
   CObject *object=object_first.Next();
   //--- borra los objetos
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Next

Coloca el puntero en el siguiente elemento de la lista.

```
void  Next(
   CObject*  object      // Puntero al siguiente elemento de la lista
   )
```

Parámetros

object

[in] Nuevo valor del puntero al siguiente elemento de la lista.

Ejemplo:

```
//--- ejemplo de CObject::Next(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Error en la creación del objeto");
      delete object_first;
      return;
     }
   //--- establece la interconexión
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utiliza los objetos
   //--- ...
   //--- borra los objetos
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Compare

Compara los datos de un ítem de la lista con los datos de otro elemento de la lista.

```
virtual int  Compare(
   const CObject*  node,      // Nodo con el que se lleva a cabo la comparación
   const int       mode=0     // Modo de comparación
   ) const
```

Parámetros

node

[in] Puntero a un ítem de la lista para hacer la comparación

mode=0

[in] Variante de la comparación

Valor devuelto

0 - si los ítemes de la lista son iguales, -1 - si el ítem de la lista es menor que el ítem de la lista de comparación (nodo), 1 - si el ítem de la lista es mayor que el ítem de la lista de comparación (nodo).

Nota

El método Compare () de la clase CObject siempre devuelve 0 y no lleva a cabo ninguna acción. Si se desea comparar datos en clases derivadas, se tiene que implementar el método Compare (...). El modo de comparación debe utilizarse cuando se implementa la comparación multivariada.

Ejemplo:

```
//--- ejemplo de CObject::Compare(...)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Error en la creación del objeto");
      delete object_first;
      return;
     }
   //--- establece la interconexión
   object_first.Next(object_second);
```

© 2000-2025, MetaQuotes Ltd.

```
   object_second.Prev(object_first);
   //--- compara objetos
   int result=object_first.Compare(object_second);
   //--- borra los objetos
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda el elemento de la lista en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen ()

Valor devuelto

true - si se ejecuta correctamente, false - en caso de error.

Nota

El método Save (int) de la clase CObject siempre devuelve true y no ejecuta ninguna acción. Si desea guardar los datos en un archivo desde una clase derivada, hay que implementar el método Save (int).

Ejemplo:

```
//--- ejemplo de CObject::Save(int)
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- establece los datos del objeto
   //--- . . .
   //--- abre el archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       FileClose(file_handle);
      }
   delete object;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los ítemes en la lista a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen ()

Valor devuelto

true - si se ejecuta correctamente, false - si hay algún error.

Nota

El método Load (int) de la clase CObject siempre devuelve true y no ejecuta ninguna acción. Si se desea cargar los datos desde un archivo en una clase derivada, se tiene que implementar el método Load (int).

Ejemplo:

```
//--- ejemplo de CObject::Load(int)
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abre el archivo
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
    }
  //--- utilizar el objeto
  //--- . . .
  delete object;
}
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo (para CObject - 0).

Ejemplo:

```
//--- ejemplo de CObject::Type()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object=new CObject;
   //---
   object=new CObject;
   if(object ==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene el identificador de tipo
   int type=object.Type();
   //--- borra el objeto
   delete object;
  }
```

© 2000-2025, MetaQuotes Ltd.

Estructuras de datos

Esta sección contiene detalles técnicos del funcionamiento de algunas estructuras de datos (arrays, listas ordenadas, etc.), así como descripciones de componentes relevantes de la Librería Estándar del lenguaje MQL5.

Las clases de estructuras de datos le ayudarán a ahorrar tiempo en el momento de crear sus propios contenedores de datos en varios formatos, incluyendo estructuras de datos compuestas.

Los conjuntos de datos de la Librería Estándar de MQL5 se encuentran en el directorio del terminal Include\Arrays.

Arrays de datos

Estas clases le permiten ahorrar tiempo en el momento de crear sus propios contenedores de datos en varios formatos, incluyendo arrays multidimensionales.

Los arrays de datos de la Librería Estándar de MQL5 se encuentran en el directorio del terminal Include\Arrays.

Clase	Descripción
CArray	Clase base de los arrays de datos dinámicos
CArrayChar	Array dinámico de variables de tipo char o uchar
CArrayShort	Array dinámico de variables de tipo short o ushort
CArrayInt	Array dinámico de variables de tipo int o uint
CArrayLong	Array dinámico de variables de tipo long o ulong
CArrayFloat	Array dinámico de variables de tipo float
CArrayDouble	Array dinámico de variables de tipo double
CArrayString	Array dinámico de variables de tipo string
CArrayObj	Array dinámico de punteros CObject
CList	Permite trabajar con una lista de instancias de CObject, y también con sus descendientes
CTreeNode	Permite trabajar con nodos del árbol binario CTree
CTree	Permite trabajar con árboles binarios de tipo CTreeNode, y también con sus descendientes

© 2000-2025, MetaQuotes Ltd.

CArray

CArray es la clase base de los arrays dinámicos de variables.

Descripción

La clase CArray permite trabajar con arrays dinámicos de variables, gestionar la memoria, ordenar los elementos, así como trabajar con archivos.

Declaración

```
class CArray : public CObject
```

Título

```
#include <Arrays\Array.mqh>
```

Jerarquía de herencia

CObject

CArray

Descendientes directos

CArrayChar, CArrayDouble, CArrayFloat, CArrayInt, CArrayLong, CArrayObj, CArrayShort, CArrayString

Métodos de la clase

Atributos	
Step	Obtiene el tamaño del incremento de paso del array.
Step	Establece el tamaño del incremento del array
Total	Obtiene el número de elementos del array
Available	Obtiene el número de elementos libres disponibles en el array sin asignación de memoria adicional
Max	Obtiene el tamaño máximo posible del array sin asignación de memoria adicional
IsSorted	Indica si el array está ordenado según la opción especificada
SortMode	Obtiene el modo de ordenación del array
Métodos de reinicio	
Clear	Borra todos los elementos del array sin liberar memoria
Métodos de ordenación	
Sort	Ordena el array con la opción especificada
Entrada/salida	

© 2000-2025, MetaQuotes Ltd.

Atributos	
virtual Save	Guarda los datos del array en el archivo
virtual Load	Carga los datos en el array a partir del archivo

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Step

Obtiene el tamaño del incremento del paso del array.

```
int  Step() const
```

Valor devuelto

Tamaño del incremento del array.

Ejemplo:

```
//--- ejemplo de CArray::Step()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene el tamaño del paso
   int step=array.Step();
   //--- utiliza el array
   //--- ...
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Step

Establece el tamaño del incremento del array.

```
bool  Step(
   int  step      // paso
   )
```

Parámetros

step

[in] El nuevo valor del incremento del paso del tamaño del array.

Valor devuelto

true si se ejecuta correctamente, false - si se ha intentado establecer un paso inferior o igual a cero.

Ejemplo:

```
//--- ejemplo de CArray::Step(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- establecer el tamaño del paso
   bool result=array.Step(1024);
   //--- utiliza el array
   //--- ...
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Total

Obtiene el número de elementos del array.

```
int  Total() const;
```

Valor devuelto

Número de elementos del array.

Ejemplo:

```
//--- ejemplo de CArray::Total()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- comprobar el total
   int total=array.Total();
   //--- utiliza el array
   //--- ...
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Available

Obtiene el número de elementos libres disponibles en el array sin asignación de memoria adicional.

```
int  Available() const
```

Valor devuelto

Número de elementos libres disponibles en el array sin asignación de memoria adicional.

Ejemplo:

```
//--- ejemplo de CArray::Available()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- comprobar disponibilidad
   int available=array.Available();
   //--- utiliza el array
   //--- ...
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Max

Obtiene el tamaño máximo posible del array sin redistribución de memoria.

```
int  Max() const
```

Valor devuelto

El tamaño máximo posible del array sin redistribución de memoria.

Ejemplo:

```
//--- ejemplo de CArray::Max()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- comprobar tamaño máximo
   int max=array.Max();
   //--- utiliza el array
   //--- ...
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

IsSorted

Indica si el array está ordenado según la opción especificada.

```
bool  IsSorted(
   int  mode=0      // Modo de ordenación
   ) const
```

Parámetros

mode=0

[in] Modo de ordenación.

Valor devuelto

Bandera de la lista ordenada. Si la lista está ordenada de acuerdo al modo especificado - true, en caso contrario - false.

Nota

La bandera de ordenación no se puede cambiar directamente. Se establece con el método Sort() y se reinicia con cualquiera de los métodos de adición e inserción a excepción del método InserSort(...).

Ejemplo:

```
//--- ejemplo de CArray::IsSorted()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- comprobar ordenación
   if(array.IsSorted())
     {
      //--- utilizar los métodos del array ordenado
      //--- ...
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SortMode

Obtiene la versión del array de ordenación.

```
int  SortMode() const;
```

Valor devuelto

Modo de ordenación.

Ejemplo:

```
//--- ejemplo de CArray::SortMode()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- comprobar el modo de ordenación
   int sort_mode=array.SortMode();
   //--- utiliza el array
   //--- ...
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

Borra todos los elementos del array sin liberar memoria.

```
void  Clear()
```

Valor devuelto

Ninguno.

Ejemplo:

```
//--- ejemplo de CArray::Clear()
#include <Arrays\Array.mqh>
//---
void OnStart()
 {
 CArray *array=new CArray;
 //---
 if(array==NULL)
 {
 printf("Error en la creación del objeto");
 return;
 }
 //--- utilizar array
 //--- ...
 //--- liberar array
 array.Clear();
 //--- borrar array
 delete array;
 }
```

© 2000-2025, MetaQuotes Ltd.

Sort

Ordena el array según la opción especificada.

```
void  Sort(
   int  mode=0       // Modo de ordenación
   )
```

Parámetros

mode=0

[in] Modo de ordenación del array.

Valor devuelto

No.

Nota

La ordenación de los arrays es siempre ascendente. En los arrays de tipos de datos primitivos (CArrayChar, CArrayShort, etc.), el parámetro mode no se utiliza. En el array CArrayObj, el orden multivariante se tiene que implementar en el método Sort (int) de la clase derivada.

Ejemplo:

```
//--- ejemplo de CArray::Sort(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- ordenación por modo 0
   array.Sort(0);
   //--- utiliza el array
   //--- ...
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los datos del array en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArray::Save(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los datos en el array a partir del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArray::Load(...)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayChar

CArrayChar es una clase de arrays dinámicos de variables de tipo char o uchar.

Descripción

La clase CArrayChar permite trabajar con arrays dinámicos de variables de tipo char o uchar. Esta clase implementa métodos para añadir, insertar y borrar elementos del array, ordenarlo, buscar elementos en el array ordenado, así como métodos para trabajar con archivos.

Declaración

```
class CArrayChar : public CArray
```

Título

```
#include <Arrays\ArrayChar.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayChar

Métodos de la clase

Control de memoria	
Reserve	Asigna memoria para incrementar el tamaño del array
Resize	Establece un nuevo tamaño, más pequeño, del array
Shutdown	Borra el array liberando toda la memoria
Métodos de adición	
Add	Añade un elemento al final del array
AddArray	Añade al final del array los elementos de otro array
AddArray	Añade al final del array los elementos de otro array
Insert	Inserta un elemento en la posición especificada
InsertArray	Inserta en la posición especificada un array de elementos de otro array
InsertArray	Inserta en la posición especificada un array de elementos de otro array
AssignArray	Copia los elementos de otro array
AssignArray	Copia los elementos de otro array
Métodos de modificación	

© 2000-2025, MetaQuotes Ltd.

Control de memoria	
Update	Cambia el elemento de la posición especificada del array
Shift	Mueve un ítem de una posición dada a la especificada por el desplazamiento
Métodos de borrado	
Delete	Borra el elemento de la posición especificada
DeleteRange	Borra el grupo de elementos de la posición especificada
Métodos de acceso	
At	Obtiene el elemento de la posición especificada
Métodos de comparación	
CompareArray	Compara el array con otro
CompareArray	Compara el array con otro
Métodos de ordenación	
InsertSort	Inserta el elemento en el array ordenado
Búsqueda	Busca el elemento en el array ordenado
SearchGreat	Busca en el array ordenado el elemento mayor que el especificado
SearchLess	Busca en el array ordenado el elemento menor que el especificado
SearchGreatOrEqual	Busca en el array ordenado el elemento mayor o igual que el especificado
SearchLessOrEqual	Busca en el array ordenado el elemento menor o igual que el especificado
SearchFirst	Busca en el array ordenado el primer elemento igual al especificado
SearchLast	Busca en el array ordenado el último elemento igual al especificado
SearchLinear	Busca en el array el elemento igual al especificado
Entrada/salida	
virtual Save	Guarda los datos del array en el archivo
virtual Load	Carga los datos en el array a partir del archivo
virtual Type	Obtiene el identificador de tipo del array

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

Asigna memoria para aumentar el tamaño del array.

```
bool  Reserve(
   int  size      // Número
   )
```

Parámetros

size

[in] Número de elementos adicionales del array.

Valor devuelto

true si se ejecuta correctamente, false - si hubo un intento de buscar una cantidad menor o igual a cero, o si el array no se incrementó.

Nota

Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayChar::Reserve(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- reservar memoria
   if(!array.Reserve(1024))
     {
      printf("Error al reservar");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Establece un tamaño nuevo, más pequeño, del array.

```
bool  Resize(
   int  size      // Tamaño
   )
```

Parámetros

size

[in] Tamaño nuevo del array.

Valor devuelto

true si se ejecuta correctamente, false - si se intenta establecer un tamaño menor a cero.

Nota

Cambiar el tamaño del array permite un uso óptimo de la memoria. Se pierden los elementos superfluos de la derecha. Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayChar::Resize(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- cambia el tamaño del array
   if(!array.Resize(10))
     {
      printf("Error al cambiar de tamaño");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Borra el array liberando por completo la memoria.

```
bool  Shutdown()
```

Valor devuelto

true si se ejecuta correctamente, false - si ocurre algún error.

Ejemplo:

```
//--- ejemplo de CArrayChar::Shutdown()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- libera el array
   if(!array.Shutdown())
     {
      printf("Error al liberar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al final del array.

```
bool  Add(
   char  element      // Elemento a añadir
   )
```

Parámetros

element

[in] valor del elemento a añadir al final del array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede añadir.

Ejemplo:

```
//--- ejemplo de CArrayChar::Add(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Error al añadir el elemento");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const char&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos a añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayChar::AddArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const CArrayChar*  src      // Puntero al array fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayChar con los elementos fuente a añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayChar::AddArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete src;
      delete array;
      return;
     }
   //--- borra el array fuente
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta un elemento en el array en la posición especificada.

```
bool  Insert(
   char  element,     // Elemento a insertar
   int   pos          // Posición
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayChar::Insert(char,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- insertar elementos
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Error al insertar");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta en la posición especificada un array de elementos a partir de otro array.

```
bool  InsertArray(
   const char&  src[],     // Array fuente
   int           pos       // Posición
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos a insertar

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayChar::InsertArray(const char &[],int)
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta en la posición especificada un array de elementos a partir de otro array.

```
bool  InsertArray(
   CArrayChar*  src,      // Pointer to the source
   int           pos      // Position
   )
```

Parámetros

src

[in] Pointer to an instance of class CArrayChar-source elements to insert.

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- example for CArrayChar::InsertArray(const CArrayChar*,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const char&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array de elementos fuente a copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayChar::AssignArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const CArrayChar*  src      // Puntero al array fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayChar con los elementos fuente a copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayChar::AssignArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayChar *src  =new CArrayChar;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete src;
      delete array;
      return;
     }
   //--- los arrays son idénticos
   //--- borra el array fuente
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Actualiza el elemento de la posición especificada del array.

```
bool  Update(
   int   pos,          // Posición
   char  element       // Valor
   )
```

Parámetros

pos

[in] Posición del elemento que se desea cambiar

element

[in] Nuevo valor del elemento

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CArrayChar::Update(int,char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- actualiza el elemento
   if(!array.Update(0,'A'))
     {
      printf("Error de actualización");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Mueve el ítem de una determinada posición según el desplazamiento especificado.

```
bool  Shift(
   int  pos,        // Posición
   int  shift       // Valor
   )
```

Parámetros

pos

[in] Posición del elemento desplazado

shift

[in] El valor del desplazamiento (positivo y negativo).

Valor devuelto

true si se ejecuta correctamente, false - si el ítem no se puede mover.

Ejemplo:

```
//--- ejemplo de CArrayChar::Shift(int,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- desplaza el elemento
   if(!array.Shift(10,-5))
     {
      printf("Error de desplazamiento");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el elemento del array en la posición especificada.

```
bool  Delete(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] La posición donde se borra el elemento del array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede borrar.

Ejemplo:

```
//--- ejemplo de CArrayChar::Delete(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra el elemento
   if(!array.Delete(0))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Borra un grupo de elementos de la posición especificada.

```
bool  DeleteRange(
   int  from,     // Posición del primer elemento
   int  to        // Posición del último elemento
   )
```

Parámetros

from

[in] Posición del primer elemento del grupo.

to

[in] Posición del último elemento del grupo.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden borrar.

Ejemplo:

```
//--- ejemplo de CArrayChar::DeleteRange(int,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra los elementos
   if(!array.DeleteRange(0,10))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del array de la posición especificada.

```
char  At(
   int  pos      // Posición
   ) const
```

Parámetros

pos

[in] Posición del elemento deseado.

Valor devuelto

El valor del elemento en caso de éxito, CHAR_MAX - si se intenta obtener un elemento en una posición que no existe (el último error ERR_OUT_OF_RANGE).

Nota

Por supuesto, CHAR_MAX puede ser un valor válido de un elemento del array, con el valor, comprobar el último código de error.

Ejemplo:

```
//--- ejemplo de CArrayChar::At(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      char result=array.At(i);
      if(result==CHAR_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- error al leer el array
         printf("Obtener el error del elemento");
         delete array;
         return;
        }
      //--- utiliza el elemento
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const char&  src[]      // Array fuente
   ) const
```

Parámetros

src[]

[in] Referencia al array que contiene los elementos fuente para llevar a cabo la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayChar::CompareArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const CArrayChar*  src      // Puntero a las fuentes
   ) const
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayChar que tiene los elementos fuente para hacer la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayChar::CompareArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra los arrays
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Inserta el elemento en el array ordenado.

```
bool  InsertSort(
   char  element      // Elemento a insertar
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array ordenado

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayChar::InsertSort(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- inserta el elemento
   if(!array.InsertSort('A'))
     {
      printf("Error de inserción");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Búsqueda

Busca un elemento igual al especificado en el array ordenado.

```
int  Search(
   char  element      // Ejemplo
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayChar::Search(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.Search('A')!=-1) printf("Elemento encontrado");
   else                      printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Busca el elemento de más muestras en el array ordenado.

```
int  SearchGreat(
   char  element      // Ejemplo
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayChar::SearchGreat(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreat('A')!=-1) printf("Elemento encontrado");
   else                           printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Busca un elemento menor que la muestra en el array ordenado.

```
int  SearchLess(
   char  element      // Ejemplo
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayChar::SearchLess(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLess('A')!=-1) printf("Elemento encontrado");
   else                          printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Busca un elemento mayor o igual a la muestra en el array ordenado.

```
int  SearchGreatOrEqual(
   char  element      // Ejemplo
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayChar::SearchGreatOrEqual(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreatOrEqual('A')!=-1) printf("Elemento encontrado");
   else                                  printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Busca un elemento menor o igual a la muestra en el array ordenado.

```
int  SearchLessOrEqual(
   char  element      // Ejemplo
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayChar::SearchLessOrEqual(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLessOrEqual('A')!=-1) printf("Elemento encontrado");
   else                                 printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Encuentra en el array ordenado el primer elemento igual a la muestra.

```
int  SearchFirst(
   char  element      // Ejemplo
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayChar::SearchFirst(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchFirst('A')!=-1) printf("Elemento encontrado");
   else                           printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Encuentra el último elemento igual a la muestra en el array ordenado.

```
int  SearchLast(
   char  element      // Ejemplo
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayChar::SearchLast(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLast('A')!=-1) printf("Elemento encontrado");
   else                          printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Busca el elemento igual a la muestra en el array.

```
int  SearchLinear(
   char  element      // Ejemplo
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 - si el elemento no se encuentra.

Nota

El método utiliza un algoritmo de búsqueda lineal (o secuencial) en arrays no ordenados.

Ejemplo:

```
//--- ejemplo de CArrayChar::SearchLinear(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- busca el elemento
   if(array.SearchLinear('A')!=-1) printf("Elemento encontrado");
   else                            printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los datos del array en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se ejecuta correctamente, false - en caso de error.

Ejemplo:

```
//--- ejemplo de CArrayChar::Save(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los datos en el array a partir del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se ejecuta correctamente, false - en caso de error.

Ejemplo:

```
//--- ejemplo de CArrayChar::Load(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilizar los elementos del array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = '%c'",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo del array.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del array (para CArrayChar, 77).

Ejemplo:

```
//--- ejemplo de CArrayChar::Type()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene el tipo del array
   int type=array.Type();
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayShort

CArrayShort proporciona un array dinámico de variables de tipo short o ushort.

Descripción

La clase CArrayShort proporciona un array dinámico de variables de tipo short o ushort. La clase permite añadir, insertar y borrar elementos, así como ordenarlos y buscarlos en un array ordenado. Además, los métodos implementados permiten trabajar con archivos.

Declaración

```
class CArrayShort : public CArray
```

Título

```
#include <Arrays\ArrayShort.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayShort

Métodos de la clase

Control de memoria	
Reserve	Asigna memoria para incrementar el tamaño del array
Resize	Establece un nuevo tamaño, más pequeño, del array
Shutdown	Borra el array liberando toda la memoria
Métodos de adición	
Add	Añade un elemento al final del array
AddArray	Añade al final del array los elementos de otro array
AddArray	Añade al final del array los elementos de otro array
Insert	Inserta un elemento en la posición especificada
InsertArray	Inserta en la posición especificada un array de elementos de otro array
InsertArray	Inserta en la posición especificada un array de elementos de otro array
AssignArray	Copia los elementos de otro array
AssignArray	Copia los elementos de otro array
Métodos de actualización	

© 2000-2025, MetaQuotes Ltd.

Control de memoria	
Update	Cambia el elemento de la posición especificada del array
Shift	Mueve un ítem de una posición dada a la especificada por el desplazamiento
Métodos de borrado	
Delete	Borra el elemento de la posición especificada
DeleteRange	Borra el grupo de elementos de la posición especificada
Métodos de acceso	
At	Obtiene el elemento de la posición especificada
Métodos de comparación	
CompareArray	Compara el array con otro
CompareArray	Compara el array con otro
Operaciones de ordenación	
InsertSort	Inserta el elemento en el array ordenado
Búsqueda	Busca en el array ordenado un elemento igual al especificado
SearchGreat	Busca en el array ordenado el elemento mayor que el especificado
SearchLess	Busca en el array ordenado el elemento menor que el especificado
SearchGreatOrEqual	Busca en el array ordenado el elemento mayor o igual que el especificado
SearchLessOrEqual	Busca en el array ordenado el elemento menor o igual que el especificado
SearchFirst	Busca en el array ordenado el primer elemento igual al especificado
SearchLast	Busca en el array ordenado el último elemento igual al especificado
SearchLinear	Busca en el array el elemento igual al especificado
Entrada/salida	
virtual Save	Guarda los datos del array en el archivo
virtual Load	Carga los datos en el array a partir del archivo
virtual Type	Obtiene el identificador de tipo del array

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

Asigna memoria para aumentar el tamaño del array.

```
bool  Reserve(
   int  size      // Número
   )
```

Parámetros

size

[in] Número de elementos adicionales del array.

Valor devuelto

true si se ejecuta correctamente, false - si hubo un intento de buscar una cantidad menor o igual a cero, o si el array no se incrementó.

Nota

Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayShort::Reserve(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- reservar memoria
   if(!array.Reserve(1024))
     {
      printf("Error al reservar");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Establece un tamaño nuevo, más pequeño, del array.

```
bool  Resize(
   int  size      // Tamaño
   )
```

Parámetros

size

[in] Tamaño nuevo del array.

Valor devuelto

true si se ejecuta correctamente, false - si se intenta establecer un tamaño menor a cero.

Nota

Cambiar el tamaño del array permite un uso óptimo de la memoria. Se pierden los elementos superfluos de la derecha. Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayShort::Resize(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- cambia el tamaño del array
   if(!array.Resize(10))
     {
      printf("Error al cambiar de tamaño");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Borra el array liberando por completo la memoria.

```
bool  Shutdown()
```

Valor devuelto

true si se ejecuta correctamente, false - si ocurre algún error.

Ejemplo:

```
//--- ejemplo de CArrayShort::Shutdown()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- libera el array
   if(!array.Shutdown())
     {
      printf("Error al liberar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al final del array.

```
bool  Add(
   short  element      // Elemento a añadir
   )
```

Parámetros

element

[in] Valor del elemento a añadir en el array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede añadir.

Ejemplo:

```
//--- ejemplo de CArrayShort::Add(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Error al añadir el elemento");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const short&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos a añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayShort::AddArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const CArrayShort*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayShort con los elementos fuente a añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayShort::AddArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete src;
      delete array;
      return;
     }
   //--- borra el array fuente
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta un elemento en el array en la posición especificada.

```
bool  Insert(
   short  element,     // Elemento a insertar
   int    pos          // Posición
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayShort::Insert(short,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- insertar elementos
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Error al insertar");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array a partir de la posición especificada.

```
bool  InsertArray(
   const short&  src[],     // Array fuente
   int           pos        // Posición
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos a insertar

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayShort::InsertArray(const short &[],int)
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array comenzando en la posición especificada.

```
bool  InsertArray(
   CArrayShort*  src,      // Puntero a la fuente
   int            pos      // Posición
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayShort con los elementos fuente para realizar la inserción.

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayShort::InsertArray(const CArrayShort*,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
      delete array;
      return;
     }
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const short&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array de elementos fuente a copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayShort::AssignArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const CArrayShort*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayShort con los elementos fuente a copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayShort::AssignArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayShort *src  =new CArrayShort;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete src;
      delete array;
      return;
     }
   //--- los arrays son idénticos
   //--- borra el array fuente
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Actualiza el elemento de la posición especificada del array.

```
bool  Update(
   int    pos,          // Posición
   short  element       // Valor
   )
```

Parámetros

pos

[in] Posición del elemento que se desea cambiar

element

[in] Nuevo valor del elemento

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CArrayShort::Update(int,short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- actualiza el elemento
   if(!array.Update(0,100))
     {
      printf("Error de actualización");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Mueve el ítem de una determinada posición según el desplazamiento especificado.

```
bool  Shift(
   int  pos,        // Posiciones
   int  shift       // Desplazamiento
   )
```

Parámetros

pos

[in] Posición del elemento desplazado

shift

[in] El valor del desplazamiento (positivo y negativo).

Valor devuelto

true si se ejecuta correctamente, false - si el ítem no se puede mover.

Ejemplo:

```
//--- ejemplo de CArrayShort::Shift(int,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- desplaza el elemento
   if(!array.Shift(10,-5))
     {
      printf("Error de desplazamiento");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el elemento del array en la posición especificada.

```
bool  Delete(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] La posición donde se borra el elemento del array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede borrar.

Ejemplo:

```
//--- ejemplo de CArrayShort::Delete(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra el elemento
   if(!array.Delete(0))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Borra un grupo de elementos de la posición especificada.

```
bool  DeleteRange(
   int  from,     // Posición del primer elemento
   int  to        // Posición del último elemento
   )
```

Parámetros

from

[in] Posición del primer elemento del grupo.

to

[in] Posición del último elemento del grupo.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden borrar.

Ejemplo:

```
//--- ejemplo de CArrayShort::DeleteRange(int,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra los elementos
   if(!array.DeleteRange(0,10))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del array de la posición especificada.

```
short  At(
   int  pos      // Posición
   ) const
```

Parámetros

pos

[in] Posición del elemento deseado.

Valor devuelto

El valor del elemento, si se ejecuta correctamente; SHORT_MAX si se intenta acceder a un elemento en una posición que no existe (el último error ERR_OUT_OF_RANGE).

Nota

Por supuesto, SHORT_MAX puede ser un valor válido de un elemento del array, de modo que, dado un valor, comprobar siempre el último código de error.

Ejemplo:

```
//--- ejemplo de CArrayShort::At(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      short result=array.At(i);
      if(result==SHORT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- error al leer el array
         printf("Obtener el error del elemento");
         delete array;
         return;
        }
      //--- utiliza el elemento
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const short&  src[]      // Array fuente
   ) const
```

Parámetros

src[]

[in] Referencia al array que contiene los elementos fuente para llevar a cabo la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayShort::CompareArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const CArrayShort*  src      // Puntero a la fuente
   ) const
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayShort con los elementos fuente para hacer la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayShort::CompareArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra los arrays
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Inserta el elemento en el array ordenado.

```
bool  InsertSort(
   short  element      // Elemento a insertar
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array ordenado

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayShort::InsertSort(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- inserta el elemento
   if(!array.InsertSort(100))
     {
      printf("Error de inserción");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Búsqueda

Busca un elemento igual al especificado en el array ordenado.

```
int  Search(
   short  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayShort::Search(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.Search(100)!=-1) printf("Elemento encontrado");
   else                      printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Busca el elemento de más muestras en el array ordenado.

```
int  SearchGreat(
   short  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayShort::SearchGreat(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreat(100)!=-1) printf("Elemento encontrado");
   else                           printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Busca un elemento menor que la muestra en el array ordenado.

```
int  SearchLess(
   short  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayShort::SearchLess(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLess(100)!=-1) printf("Elemento encontrado");
   else                          printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Busca en el array ordenado un elemento mayor o igual a la muestra.

```
int  SearchGreatOrEqual(
   short  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayShort::SearchGreatOrEqual(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreatOrEqual(100)!=-1) printf("Elemento encontrado");
   else                                  printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Busca un elemento menor o igual a la muestra en el array ordenado.

```
int  SearchLessOrEqual(
   short  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayShort::SearchLessOrEqual(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLessOrEqual(100)!=-1) printf("Elemento encontrado");
   else                                 printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Encuentra en el array ordenado el primer elemento igual a la muestra.

```
int  SearchFirst(
   short  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayShort::SearchFirst(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchFirst(100)!=-1) printf("Elemento encontrado");
   else                           printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Encuentra el último elemento igual a la muestra en el array ordenado.

```
int  SearchLast(
   short  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayShort::SearchLast(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLast(100)!=-1) printf("Elemento encontrado");
   else                          printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Busca el elemento igual a la muestra en el array.

```
int  SearchLinear(
   short  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 - si el elemento no se encuentra.

Nota

El método utiliza un algoritmo de búsqueda lineal (o secuencial) en arrays no ordenados.

Ejemplo:

```
//--- ejemplo de CArrayShort::SearchLinear(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- busca el elemento
   if(array.SearchLinear(100)!=-1) printf("Elemento encontrado");
   else                            printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los datos del array en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayShort::Save(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade 100 elementos al array
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los datos en el array a partir del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayShort::Load(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilizar los elementos del array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = %d",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo del array.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del array (el de CArrayShort es 82).

Ejemplo:

```
//--- ejemplo de CArrayShort::Type()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene el tipo del array
   int type=array.Type();
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayInt

La clase CArrayInt proporciona un array dinámico de variables de tipo int o uint.

Descripción

La clase CArrayInt permite trabajar con un array dinámico de variables de tipo int o uint. Esta clase implementa métodos para añadir, insertar y borrar elementos de un array, así como buscarlos y ordenarlos. Además, los métodos permiten trabajar con archivos.

Declaración

```
class CArrayInt : public CArray
```

Título

```
#include <Arrays\ArrayInt.mqh>
```

Jerarquía de herencia

CObject

CArray

CArrayInt

Descendientes directos

CSpreadBuffer

Métodos de la clase

Control de memoria	
Reserve	Asigna memoria para incrementar el tamaño del array
Resize	Establece un nuevo tamaño, más pequeño, del array
Shutdown	Borra el array liberando toda la memoria
Métodos de adición	
Add	Añade un elemento al final del array
AddArray	Añade al final del array los elementos de otro array
AddArray	Añade al final del array los elementos de otro array
Insert	Inserta un elemento en la posición especificada
InsertArray	Inserta en la posición especificada un array de elementos de otro array
InsertArray	Inserta en la posición especificada un array de elementos de otro array
AssignArray	Copia los elementos de otro array

© 2000-2025, MetaQuotes Ltd.

Control de memoria	
AssignArray	Copia los elementos de otro array
Métodos de actualización	
Update	Cambia el elemento de la posición especificada del array
Shift	Mueve un ítem de una posición dada a la especificada por el desplazamiento
Métodos de borrado	
Delete	Borra el elemento de la posición especificada
DeleteRange	Borra el grupo de elementos de la posición especificada
Métodos de acceso	
At	Obtiene el elemento de la posición especificada
Métodos de comparación	
CompareArray	Compara el array con otro
CompareArray	Compara el array con otro
Operaciones de ordenación	
InsertSort	Inserta el elemento en el array ordenado
Búsqueda	Busca en el array ordenado un elemento igual al especificado
SearchGreat	Busca en el array ordenado el elemento mayor que el especificado
SearchLess	Busca en el array ordenado el elemento menor que el especificado
SearchGreatOrEqual	Busca en el array ordenado el elemento mayor o igual que el especificado
SearchLessOrEqual	Busca en el array ordenado el elemento menor o igual que el especificado
SearchFirst	Busca en el array ordenado el primer elemento igual al especificado
SearchLast	Busca en el array ordenado el último elemento igual al especificado
SearchLinear	Busca en el array el elemento igual al especificado
Entrada/salida	
virtual Save	Guarda los datos del array en el archivo
virtual Load	Carga los datos en el array a partir del archivo
virtual Type	Obtiene el identificador de tipo del array

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

Asigna memoria para aumentar el tamaño del array.

```
bool  Reserve(
   int  size      // Número
   )
```

Parámetros

size

[in] Número de elementos adicionales del array.

Valor devuelto

true si se ejecuta correctamente, false - si hubo un intento de buscar una cantidad menor o igual a cero, o si el array no se incrementó.

Nota

Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayInt::Reserve(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- reservar memoria
   if(!array.Reserve(1024))
     {
      printf("Error al reservar");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Establece un tamaño nuevo, más pequeño, del array.

```
bool  Resize(
   int  size      // Número
   )
```

Parámetros

size

[in] Tamaño nuevo del array.

Valor devuelto

true si se ejecuta correctamente, false - si se intenta establecer un tamaño menor a cero.

Nota

Cambiar el tamaño del array permite un uso óptimo de la memoria. Se pierden los elementos superfluos de la derecha. Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayInt::Resize(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- cambia el tamaño del array
   if(!array.Resize(10))
     {
      printf("Error al cambiar de tamaño");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Borra el array liberando por completo la memoria.

```
bool  Shutdown()
```

Valor devuelto

true si se ejecuta correctamente, false - si ocurre algún error.

Ejemplo:

```
//--- ejemplo de CArrayInt::Shutdown()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- libera el array
   if(!array.Shutdown())
     {
      printf("Error al liberar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al final del array.

```
bool  Add(
   int  element      // Elemento a añadir
   )
```

Parámetros

element

[in] valor del elemento a añadir al final del array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede añadir.

Ejemplo:

```
//--- ejemplo de CArrayInt::Add(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Error al añadir el elemento");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const int&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos a añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayInt::AddArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const CArrayInt*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayInt con los elementos fuente que se desean añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayInt::AddArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete src;
      delete array;
      return;
     }
   //--- borra el array fuente
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta un elemento en el array en la posición especificada.

```
bool  Insert(
   int  element,     // Elemento a insertar
   int  pos          // Posición
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayInt::Insert(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- insertar elementos
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Error al insertar");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array a partir de la posición especificada.

```
bool  InsertArray(
   const int&  src[],     // Array fuente
   int          pos        // Posición
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos a insertar

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayInt::InsertArray(const int &[],int)
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array a partir de la posición especificada.

```
bool  InsertArray(
   CArrayInt*  src,     // Puntero a la fuente
   int          pos      // Posición
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayInt con los elementos fuente que se desean insertar.

pos

[in] Posición a insertar en el array.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayInt::InsertArray(const CArrayInt*,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const int&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array de elementos fuente a copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayInt::AssignArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const CArrayInt*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayInt con los elementos fuente que se desean copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayInt::AssignArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayInt *src  =new CArrayInt;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete src;
      delete array;
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- los arrays son idénticos
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Actualiza el elemento de la posición especificada del array.

```
bool  Update(
   int  pos,          // Posición
   int  element       // Valor
   )
```

Parámetros

pos

[in] Posición del elemento que se desea cambiar.

element

[in] Nuevo valor del elemento

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CArrayInt::Update(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- actualiza el elemento
   if(!array.Update(0,10000))
     {
      printf("Error de actualización");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Mueve el ítem de una determinada posición según el desplazamiento especificado.

```
bool  Shift(
   int  pos,        // Posición
   int  shift       // Desplazamiento
   )
```

Parámetros

pos

[in] Posición del elemento desplazado

shift

[in] El valor del desplazamiento (positivo y negativo).

Valor devuelto

true si se ejecuta correctamente, false - si el ítem no se puede mover.

Ejemplo:

```
//--- ejemplo de CArrayInt::Shift(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- desplaza el elemento
   if(!array.Shift(10,-5))
     {
      printf("Error de desplazamiento");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el elemento del array en la posición especificada.

```
bool  Delete(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] La posición donde se borra el elemento del array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede borrar.

Ejemplo:

```
//--- ejemplo de CArrayInt::Delete(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra el elemento
   if(!array.Delete(0))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Borra un grupo de elementos de la posición especificada.

```
bool  DeleteRange(
   int  from,     // Posición del primer elemento
   int  to        // Posición del último elemento
   )
```

Parámetros

from

[in] Posición del primer elemento del grupo.

to

[in] Posición del último elemento del grupo.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden borrar.

Ejemplo:

```
//--- ejemplo de CArrayInt::DeleteRange(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra los elementos
   if(!array.DeleteRange(0,10))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del array de la posición especificada.

```
int  At(
   int  pos      // Posición
   ) const
```

Parámetros

pos

[in] Posición del elemento deseado.

Valor devuelto

Si se ejecuta correctamente devuelve el valor del elemento; en caso contrario, devuelve INT_MAX si se intenta acceder a un elemento en una posición que no existe (el último error ERR_OUT_OF_RANGE).

Nota

Por supuesto, INT_MAX puede ser un valor válido de un elemento del array, de modo que, dado un valor, comprobar siempre el último código de error.

Ejemplo:

```
//--- ejemplo de CArrayInt::At(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      int result=array.At(i);
      if(result==INT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- error al leer el array
         printf("Obtener el error del elemento");
         delete array;
         return;
        }
      //--- utiliza el elemento
      //--- . . .
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const int&  src[]      // Array fuente
   ) const
```

Parámetros

src[]

[in] Referencia al array que contiene los elementos fuente para llevar a cabo la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayInt::CompareArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const CArrayInt*  src      // Puntero a la fuente
   ) const
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayInt que contiene los elementos fuente para hacer la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayInt::CompareArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra los arrays
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Inserta el elemento en el array ordenado.

```
bool  InsertSort(
   int  element      // Elemento a insertar
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array ordenado

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayInt::InsertSort(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- inserta el elemento
   if(!array.InsertSort(10000))
     {
      printf("Error de inserción");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Búsqueda

Busca un elemento igual al especificado en el array ordenado.

```
int  Search(
   int  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayInt::Search(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.Search(10000)!=-1) printf("Elemento encontrado");
   else                        printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Busca el elemento de más muestras en el array ordenado.

```
int  SearchGreat(
   int  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayInt::SearchGreat(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreat(10000)!=-1) printf("Elemento encontrado");
   else                             printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Busca un elemento menor que la muestra en el array ordenado.

```
int  SearchLess(
   int  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayInt::SearchLess(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLess(10000)!=-1) printf("Elemento encontrado");
   else                            printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Busca en el array ordenado un elemento mayor o igual a la muestra.

```
int  SearchGreatOrEqual(
   int  element      // Elemento a buscar
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayInt::SearchGreatOrEqual(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreatOrEqual(10000)!=-1) printf("Elemento encontrado");
   else                                    printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Busca un elemento menor o igual a la muestra en el array ordenado.

```
int  SearchLessOrEqual(
   int  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayInt::SearchLessOrEqual(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLessOrEqual(10000)!=-1) printf("Elemento encontrado");
   else                                   printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Encuentra en el array ordenado el primer elemento igual a la muestra.

```
int  SearchFirst(
   int  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayInt:: SearchFirst(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchFirst(10000)!=-1) printf("Elemento encontrado");
   else                             printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Encuentra el último elemento igual a la muestra en el array ordenado.

```
int  SearchLast(
   int  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayInt::SearchLast(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLast(10000)!=-1) printf("Elemento encontrado");
   else                            printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Busca el elemento igual a la muestra en el array.

```
int  SearchLinear(
   int  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 - si el elemento no se encuentra.

Nota

El método utiliza un algoritmo de búsqueda lineal (o secuencial) en arrays no ordenados.

Ejemplo:

```
//--- ejemplo de CArrayInt::SearchLinear(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- busca el elemento
   if(array.SearchLinear(10000)!=-1) printf("Elemento encontrado");
   else                              printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los datos del array en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayInt::Save(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade 100 elementos al array
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los datos en el array a partir del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayInt::Load(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilizar los elementos del array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = %d",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo del array.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del array (el de CArrayInt es 82).

Ejemplo:

```
//--- ejemplo de CArrayInt::Type()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene el tipo del array
   int type=array.Type();
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayLong

La clase CArrayLong proporciona un array de variables de tipo long o ulong.

Descripción

La clase CArrayLong permite trabajar con arrays dinámicos de variables de tipo long o ulong. Esta clase implementa métodos para añadir, insertar y borrar elementos del array, así como ordenarlo y buscar elementos en el array ordenado. Además, los métodos implementados permiten trabajar con archivos.

Declaración

```
class CArrayLong : public CArray
```

Título

```
#include <Arrays\ArrayLong.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayLong

Descendientes directos

CRealVolumeBuffer, CTickVolumeBuffer, CTimeBuffer

Métodos de la clase

Control de memoria	
Reserve	Asigna memoria para incrementar el tamaño del array
Resize	Establece un nuevo tamaño, más pequeño, del array
Shutdown	Borra el array liberando toda la memoria
Métodos de adición	
Add	Añade un elemento al final del array
AddArray	Añade al final del array los elementos de otro array
AddArray	Añade al final del array los elementos de otro array
Insert	Inserta un elemento en la posición especificada
InsertArray	Inserta en la posición especificada un array de elementos de otro array
InsertArray	Inserta en la posición especificada un array de elementos de otro array
AssignArray	Copia los elementos de otro array

© 2000-2025, MetaQuotes Ltd.

Control de memoria	
AssignArray	Copia los elementos de otro array
Métodos de actualización	
Update	Cambia el elemento de la posición especificada del array
Shift	Mueve un ítem de una posición dada a la especificada por el desplazamiento
Métodos de borrado	
Delete	Borra el elemento de la posición especificada
DeleteRange	Borra el grupo de elementos de la posición especificada
Métodos de acceso	
At	Obtiene el elemento de la posición especificada
Métodos de comparación	
CompareArray	Compara el array con otro
CompareArray	Compara el array con otro
Operaciones de ordenación	
InsertSort	Inserta el elemento en el array ordenado
Búsqueda	Busca en el array ordenado un elemento igual al especificado
SearchGreat	Busca en el array ordenado el elemento mayor que el especificado
SearchLess	Busca en el array ordenado el elemento menor que el especificado
SearchGreatOrEqual	Busca en el array ordenado el elemento mayor o igual que el especificado
SearchLessOrEqual	Busca en el array ordenado el elemento menor o igual que el especificado
SearchFirst	Busca en el array ordenado el primer elemento igual al especificado
SearchLast	Busca en el array ordenado el último elemento igual al especificado
SearchLinear	Busca en el array el elemento igual al especificado
Entrada/salida	
virtual Save	Guarda los datos del array en el archivo
virtual Load	Carga los datos en el array a partir del archivo
virtual Type	Obtiene el identificador de tipo del array

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

Asigna memoria para aumentar el tamaño del array.

```
bool  Reserve(
   int  size      // Número
   )
```

Parámetros

size

[in] Número de elementos adicionales del array.

Valor devuelto

true si se ejecuta correctamente, false - si hubo un intento de buscar una cantidad menor o igual a cero, o si el array no se incrementó.

Nota

Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayLong::Reserve(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- reservar memoria
   if(!array.Reserve(1024))
     {
      printf("Error al reservar");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Establece un tamaño nuevo, más pequeño, del array.

```
bool  Resize(
   int  size      // Tamaño
   )
```

Parámetros

size

[in] Tamaño nuevo del array.

Valor devuelto

true si se ejecuta correctamente, false - si se intenta establecer un tamaño menor a cero.

Nota

Cambiar el tamaño del array permite un uso óptimo de la memoria. Se pierden los elementos superfluos de la derecha. Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayLong::Resize(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- cambia el tamaño del array
   if(!array.Resize(10))
     {
      printf("Error al cambiar de tamaño");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Borra el array liberando por completo la memoria.

```
bool  Shutdown()
```

Valor devuelto

true si se ejecuta correctamente, false - si ocurre algún error.

Ejemplo:

```
//--- ejemplo de CArrayLong::Shutdown()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- libera el array
   if(!array.Shutdown())
     {
      printf("Error al liberar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al final del array.

```
bool  Add(
   long  element      // Elemento a añadir
   )
```

Parámetros

element

[in] Valor del elemento a añadir en el array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede añadir.

Ejemplo:

```
//--- ejemplo de CArrayLong::Add(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Error al añadir el elemento");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const long&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos a añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayLong::AddArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const CArrayLong*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayLong con los elementos fuente a añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayLong::AddArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete src;
      delete array;
      return;
     }
   //--- borra el array fuente
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta un elemento en el array en la posición especificada.

```
bool  Insert(
   long  element,     // Elemento a insertar
   int   pos          // Posición
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayLong::Insert(long,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- insertar elementos
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Error al insertar");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array comenzando en la posición especificada.

```
bool  InsertArray(
   const long&  src[],     // Array fuente
   int           pos        // Posición
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos a insertar

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayLong::InsertArray(const long &[],int)
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array comenzando en la posición especificada.

```
bool  InsertArray(
   CArrayLong*  src,      // Puntero a la fuente
   int           pos      // Posición
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayLong con los elementos fuente a insertar.

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayLong::InsertArray(const CArrayLong*,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const long&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array de elementos fuente a copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayLong::AssignArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const CArrayLong*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayLong con los elementos fuente que se desean copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayLong::AssignArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayLong *src  =new CArrayLong;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete src;
      delete array;
      return;
     }
   //--- los arrays son idénticos
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Actualiza el elemento de la posición especificada del array.

```
bool  Update(
   int   pos,          // Posición
   long  element       // Valor
   )
```

Parámetros

pos

[in] Posición del elemento que se desea cambiar

element

[in] Nuevo valor del elemento

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CArrayLong::Update(int,long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- actualiza el elemento
   if(!array.Update(0,1000000))
     {
      printf("Error de actualización");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Mueve el ítem de una determinada posición según el desplazamiento especificado.

```
bool  Shift(
   int  pos,        // Posición
   int  shift       // Desplazamiento
   )
```

Parámetros

pos

[in] Posición del elemento desplazado

shift

[in] El valor del desplazamiento (positivo y negativo).

Valor devuelto

true si se ejecuta correctamente, false - si el ítem no se puede mover.

Ejemplo:

```
//--- ejemplo de CArrayLong::Shift(int,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- desplaza el elemento
   if(!array.Shift(10,-5))
     {
      printf("Error de desplazamiento");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el elemento del array en la posición especificada.

```
bool  Delete(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] La posición donde se borra el elemento del array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede borrar.

Ejemplo:

```
//--- ejemplo de CArrayLong::Delete(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra el elemento
   if(!array.Delete(0))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Borra un grupo de elementos de la posición especificada.

```
bool  DeleteRange(
   int  from,     // Posición del primer elemento
   int  to        // Posición del último elemento
   )
```

Parámetros

from

[in] Posición del primer elemento del grupo.

to

[in] Posición del último elemento del grupo.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden borrar.

Ejemplo:

```
//--- ejemplo de CArrayLong::DeleteRange(int,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra los elementos
   if(!array.DeleteRange(0,10))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del array de la posición especificada.

```
long  At(
   int  pos      // Posición
   ) const
```

Parámetros

pos

[in] Posición del elemento deseado.

Valor devuelto

El valor del elemento en caso de éxito, LONG_MAX si se intenta acceder a un elemento en una posición que no existe (el último error ERR_OUT_OF_RANGE).

Nota

Por supuesto, LONG_MAX puede ser un valor válido de un elemento del array, de modo que, dado un valor, comprobar siempre el último código de error.

Ejemplo:

```
//--- ejemplo de CArrayLong::At(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      long result=array.At(i);
      if(result==LONG_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- Error al leer el array
         printf("Obtener el error del elemento");
         delete array;
         return;
        }
      //--- utiliza el elemento
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const long&  src[]      // Array fuente
   ) const
```

Parámetros

src[]

[in] Referencia al array que contiene los elementos fuente para llevar a cabo la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayLong::CompareArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArrayconst

Compara el array con otro array.

```
bool  CompareArrayconst(
   const CArrayLong*  src      // Puntero a la fuente
   ) const
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayLong con elementos fuente para hacer la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayLong::CompareArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra los arrays
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Inserta el elemento en el array ordenado.

```
bool  InsertSort(
   long  element      // Elemento a insertar
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array ordenado

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayLong::InsertSort(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- inserta el elemento
   if(!array.InsertSort(1000000))
     {
      printf("Error de inserción");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Búsqueda

Busca el elemento igual a la muestra en el array ordenado.

```
int  Search(
   long  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayLong::Search(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.Search(1000000)!=-1) printf("Elemento encontrado");
   else                          printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Busca el elemento mayor a la muestra en el array ordenado.

```
int  SearchGreat(
   long  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayLong::SearchGreat(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreat(1000000)!=-1) printf("Elemento encontrado");
   else                               printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Busca un elemento menor que la muestra en el array ordenado.

```
int  SearchLess(
   long  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayLong::SearchLess(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLess(1000000)!=-1) printf("Elemento encontrado");
   else                              printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Busca un elemento mayor o igual a la muestra en el array ordenado.

```
int  SearchGreatOrEqual(
   long  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayLong::SearchGreatOrEqual(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreatOrEqual(1000000)!=-1) printf("Elemento encontrado");
   else                                      printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Busca el elemento menor o igual a la muestra en el array ordenado.

```
int  SearchLessOrEqual(
   long  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayLong::SearchLessOrEqual(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLessOrEqual(1000000)!=-1) printf("Elemento encontrado");
   else                                     printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Encuentra el primer elemento igual a la muestra en el array ordenado.

```
int  SearchFirst(
   long  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayLong::SearchFirst(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchFirst(1000000)!=-1) printf("Elemento encontrado");
   else                               printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Encuentra el último elemento igual a la muestra en el array ordenado.

```
int  SearchLast(
   long  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayLong::SearchLast(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLast(1000000)!=-1) printf("Elemento encontrado");
   else                              printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Busca el elemento igual a la muestra en el array.

```
int  SearchLinear(
   long  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 - si el elemento no se encuentra.

Nota

El método utiliza un algoritmo de búsqueda lineal (o secuencial) en arrays no ordenados.

Ejemplo:

```
//--- ejemplo de CArrayLong::SearchLinear(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- busca el elemento
   if(array.SearchLinear(1000000)!=-1) printf("Elemento encontrado");
   else                                printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los datos del array en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se ejecuta correctamente, false - en caso de error.

Ejemplo:

```
//--- ejemplo de CArrayLong::Save(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade 100 elementos al array
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los datos en el array a partir del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se ejecuta correctamente, false - en caso de error.

Ejemplo:

```
//--- ejemplo de CArrayLong::Load(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilizar los elementos del array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = %I64",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo del array.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del array (el de CArrayLong es 84).

Ejemplo:

```
//--- ejemplo de CArrayLong::Type()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene el tipo del array
   int type=array.Type();
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayFloat

CArrayFloat es una clase que proporciona un array dinámico de variables de tipo float.

Descripción

La clase CArrayFloat permite trabajar con un array dinámico de variables de tipo float. Con esta clase se pueden añadir, insertar y borrar elementos, así como ordenarlos y buscarlos. Además, los métodos implementados permiten trabajar con archivos.

Declaración

```
class CArrayFloat : public CArray
```

Título

```
#include <Arrays\ArrayFloat.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayFloat

Métodos de la clase

Atributos	
Delta	Establece la tolerancia de comparación
Control de memoria	
Reserve	Asigna memoria para incrementar el tamaño del array
Resize	Establece un nuevo tamaño, más pequeño, del array
Shutdown	Borra el array liberando toda la memoria
Métodos de adición	
Add	Añade un elemento al final del array
AddArray	Añade al final del array los elementos de otro array
AddArray	Añade al final del array los elementos de otro array
Insert	Inserta un elemento en la posición especificada
InsertArray	Inserta en la posición especificada un array de elementos de otro array
InsertArray	Inserta en la posición especificada un array de elementos de otro array
AssignArray	Copia los elementos de otro array

© 2000-2025, MetaQuotes Ltd.

Atributos	
AssignArray	Copia los elementos de otro array
Métodos de actualización	
Update	Cambia el elemento de la posición especificada del array
Shift	Mueve un ítem de una posición dada a la especificada por el desplazamiento
Delete	Borra el elemento de la posición especificada
DeleteRange	Borra el grupo de elementos de la posición especificada
Métodos de acceso	
At	Obtiene el elemento de la posición especificada
Métodos de comparación	
CompareArray	Compara el array con otro
CompareArray	Compara el array con otro
Operaciones de ordenación	
InsertSort	Inserta el elemento en el array ordenado
Búsqueda	Busca en el array ordenado un elemento igual al especificado
SearchGreat	Busca en el array ordenado el elemento mayor que el especificado
SearchLess	Busca en el array ordenado el elemento menor que el especificado
SearchGreatOrEqual	Busca en el array ordenado el elemento mayor o igual que el especificado
SearchLessOrEqual	Busca en el array ordenado el elemento menor o igual que el especificado
SearchFirst	Busca en el array ordenado el primer elemento igual al especificado
SearchLast	Busca en el array ordenado el último elemento igual al especificado
SearchLinear	Busca en el array el elemento igual al especificado
Entrada/salida	
virtual Save	Guarda los datos del array en el archivo
virtual Load	Carga los datos en el array a partir del archivo
virtual Type	Obtiene el identificador de tipo del array

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Delta

Establece la tolerancia de comparación.

```
void  Delta(
   float  delta      // Tolerancia
   )
```

Parámetros

delta

[in] El valor nuevo de la tolerancia de comparación.

Valor devuelto

Ninguno

Nota

Tolerancia de comparación utilizada en la búsqueda. Los valores se consideran iguales si su diferencia es menor o igual a la tolerancia. La tolerancia predeterminada es 0.0.

Ejemplo:

```
//--- ejemplo de CArrayFloat::Delta(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- establece la tolerancia de comparación
   array.Delta(0.001);
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

Asigna memoria para aumentar el tamaño del array.

```
bool  Reserve(
   int  size      // Número
   )
```

Parámetros

size

[in] Número de elementos adicionales del array.

Valor devuelto

true si se ejecuta correctamente, false - si hubo un intento de buscar una cantidad menor o igual a cero, o si el array no se incrementó.

Nota

Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayFloat::Reserve(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- reservar memoria
   if(!array.Reserve(1024))
     {
      printf("Error al reservar");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Establece un tamaño nuevo, más pequeño, del array.

```
bool  Resize(
   int  size      // Tamaño
   )
```

Parámetros

size

[in] Tamaño nuevo del array.

Valor devuelto

true si se ejecuta correctamente, false - si se intenta establecer un tamaño menor a cero.

Nota

Cambiar el tamaño del array permite un uso óptimo de la memoria. Se pierden los elementos superfluos de la derecha. Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayFloat::Resize(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- cambia el tamaño del array
   if(!array.Resize(10))
     {
      printf("Error al cambiar de tamaño");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Borra el array liberando por completo la memoria.

```
bool  Shutdown()
```

Valor devuelto

true si se ejecuta correctamente, false - si ocurre algún error.

Ejemplo:

```
//--- ejemplo de CArrayFloat::Shutdown()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- libera el array
   if(!array.Shutdown())
     {
      printf("Error al liberar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al final del array.

```
bool  Add(
   float  element      // Elemento a añadir
   )
```

Parámetros

element

[in] Valor del elemento a añadir en el array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede añadir.

Ejemplo:

```
//--- ejemplo de CArrayFloat::Add(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Error al añadir el elemento");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const float&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos a añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayFloat::AddArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const CArrayFloat*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayFloat con los elementos fuente que se desean añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayFloat::AddArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete src;
      delete array;
      return;
     }
   //--- borra el array fuente
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta un elemento en el array en la posición especificada.

```
bool  Insert(
   float  element,     // Elemento a insertar
   int    pos          // Posición
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayFloat::Insert(float,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- insertar elementos
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Error al insertar");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta en la posición especificada un array de elementos a partir de otro array.

```
bool  InsertArray(
   const float&  src[],     // Array fuente
   int           pos        // Posición
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos a insertar

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayFloat::InsertArray(const float &[],int)
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array a partir de la posición especificada.

```
bool  InsertArray(
   CArrayFloat*  src,     // Puntero a la fuente
   int            pos      // Posición
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayFloat con los elementos fuente a insertar.

pos

[in] Posición del array donde se realiza la inserción

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayFloat::InsertArray(const CArrayFloat*,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const float&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array de elementos fuente a copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayFloat::AssignArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const CArrayFloat*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayFloat con los elementos fuente que se desean copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayFloat::AssignArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayFloat *src  =new CArrayFloat;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete src;
      delete array;
      return;
     }
   //--- los arrays son idénticos
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Actualiza el elemento de la posición especificada del array.

```
bool  Update(
   int    pos,          // Posición
   float  element       // Valor
   )
```

Parámetros

pos

[in] Posición del elemento que se desea cambiar

element

[in] Nuevo valor del elemento

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CArrayFloat::Update(int,float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- actualiza el elemento
   if(!array.Update(0,100.0))
     {
      printf("Error de actualización");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Mueve el ítem de una determinada posición según el desplazamiento especificado.

```
bool  Shift(
   int  pos,        // Posición
   int  shift       // Desplazamiento
   )
```

Parámetros

pos

[in] Posición del elemento desplazado

shift

[in] El valor del desplazamiento (positivo y negativo).

Valor devuelto

true si se ejecuta correctamente, false - si el ítem no se puede mover.

Ejemplo:

```
//--- ejemplo de CArrayFloat::Shift(int,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- desplaza el elemento
   if(!array.Shift(10,-5))
     {
      printf("Error de desplazamiento");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el elemento del array en la posición especificada.

```
bool  Delete(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] La posición donde se borra el elemento del array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede borrar.

Ejemplo:

```
//--- ejemplo de CArrayFloat::Delete(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra el elemento
   if(!array.Delete(0))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Borra un grupo de elementos de la posición especificada.

```
bool  DeleteRange(
   int  from,     // Posición del primer elemento
   int  to        // Posición del último elemento
   )
```

Parámetros

from

[in] Posición del primer elemento del grupo.

to

[in] Posición del último elemento del grupo.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden borrar.

Ejemplo:

```
//--- ejemplo de CArrayFloat::DeleteRange(int,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra los elementos
   if(!array.DeleteRange(0,10))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del array de la posición especificada.

```
float  At(
   int  pos      // Posición
   ) const
```

Parámetros

pos

[in] Posición del elemento deseado.

Valor devuelto

El valor del elemento si se ejecuta correctamente, FLT_MAX si se intenta acceder a un elemento de una posición que no existe (el último error ERR_OUT_OF_RANGE).

Nota

Por supuesto, FLT_MAX puede ser un valor válido de un elemento del array, de modo que, dado un valor, comprobar siempre el último código de error.

Ejemplo:

```
//--- ejemplo de CArrayFloat::At(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      float result=array.At(i);
      if(result==FLT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- error al leer el array
         printf("Obtener el error del elemento");
         delete array;
         return;
        }
      //--- utiliza el elemento
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const float&  src[]      // Array fuente
   ) const
```

Parámetros

src[]

[in] Referencia al array que contiene los elementos fuente para llevar a cabo la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayFloat::CompareArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArrayconst

Copia los elementos de otro array.

```
bool  AssignArrayconst(
   const CArrayFloat*  src      // Puntero a la fuente
   ) const
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayFloat con los elementos fuente que se desean copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayFloat::CompareArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra los arrays
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Inserta el elemento en el array ordenado.

```
bool  InsertSort(
   float  element      // Elemento a insertar
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array ordenado

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayFloat::InsertSort(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- inserta el elemento
   if(!array.InsertSort(100.0))
     {
      printf("Error de inserción");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Búsqueda

Busca un elemento igual al especificado en el array ordenado.

```
int  Search(
   float  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayFloat::Search(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.Search(100.0)!=-1) printf("Elemento encontrado");
   else                        printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Busca el elemento de más muestras en el array ordenado.

```
int  SearchGreat(
   float  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayFloat::SearchGreat(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreat(100.0)!=-1) printf("Elemento encontrado");
   else                             printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Busca un elemento menor que la muestra en el array ordenado.

```
int  SearchLess(
   float  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayFloat:: SearchLess(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLess(100.0)!=-1) printf("Elemento encontrado");
   else                            printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Busca en el array ordenado un elemento mayor o igual a la muestra.

```
int  SearchGreatOrEqual(
   float  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayFloat::SearchGreatOrEqual(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Elemento encontrado");
   else                                    printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Busca un elemento menor o igual a la muestra en el array ordenado.

```
int  SearchLessOrEqual(
   float  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayFloat::SearchLessOrEqual(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Elemento encontrado");
   else                                   printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Encuentra en el array ordenado el primer elemento igual a la muestra.

```
int  SearchFirst(
   float  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayFloat::SearchFirst(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchFirst(100.0)!=-1) printf("Elemento encontrado");
   else                             printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Encuentra el último elemento igual a la muestra en el array ordenado.

```
int  SearchLast(
   float  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayFloat::SearchLast(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLast(100.0)!=-1) printf("Elemento encontrado");
   else                            printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Busca el elemento igual a la muestra en el array.

```
int  SearchLinear(
   float  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 - si el elemento no se encuentra.

Nota

El método utiliza un algoritmo de búsqueda lineal (o secuencial) en arrays no ordenados.

Ejemplo:

```
//--- ejemplo de CArrayFloat::SearchLinear(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- busca el elemento
   if(array.SearchLinear(100.0)!=-1) printf("Elemento encontrado");
   else                              printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los datos del array en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayFloat::Save(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade 100 elementos al array
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los datos en el array a partir del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayFloat::Load(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilizar los elementos del array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = %f",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo del array.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del array (el de CArrayFloat es 87).

Ejemplo:

```
//--- ejemplo de CArrayFloat::Type()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene el tipo del array
   int type=array.Type();
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayDouble

CArrayDouble es una clase que proporciona un array dinámico de variables de tipo double.

Descripción

La clase CArrayDouble permite trabajar con un array dinámico de variables de tipo double. La clase implementa métodos para añadir elementos, insertarlos y borrarlos, así como métodos de ordenación y búsqueda de elementos. Además, los métodos implementados pueden trabajar con archivos.

Declaración

```
class CArrayDouble : public CArray
```

Título

```
#include <Arrays\ArrayDouble.mqh>
```

Jerarquía de herencia

CObject

CArray

CArrayDouble

Descendientes directos

CDoubleBuffer

Métodos de la clase

Atributos	
Delta	Establece la tolerancia de comparación
Control de memoria	
Reserve	Asigna memoria para incrementar el tamaño del array
Resize	Establece un nuevo tamaño, más pequeño, del array
Shutdown	Borra el array liberando toda la memoria
Métodos de adición	
Add	Añade un elemento al final del array
AddArray	Añade al final del array los elementos de otro array
AddArray	Añade al final del array los elementos de otro array
Insert	Inserta un elemento en la posición especificada
InsertArray	Inserta en la posición especificada un array de elementos de otro array

© 2000-2025, MetaQuotes Ltd.

Atributos	
InsertArray	Inserta en la posición especificada un array de elementos de otro array
AssignArray	Copia los elementos de otro array
AssignArray	Copia los elementos de otro array
Métodos de actualización	
Update	Cambia el elemento de la posición especificada del array
Shift	Mueve un ítem de una posición dada a la especificada por el desplazamiento
Métodos de borrado	
Delete	Borra el elemento de la posición especificada
DeleteRange	Borra el grupo de elementos de la posición especificada
Métodos de acceso	
At	Obtiene el elemento de la posición especificada
Métodos de comparación	
CompareArray	Compara el array con otro
CompareArray	Compara el array con otro
Búsqueda de valores mínimo/máximo	
Minimum	Obtiene el índice del elemento más pequeño del array, en el intervalo especificado
Maximum	Obtiene el índice del elemento más grande del array, en el intervalo especificado.
Operaciones de ordenación	
InsertSort	Inserta el elemento en el array ordenado
Búsqueda	Busca en el array ordenado un elemento igual al especificado
SearchGreat	Busca en el array ordenado el elemento mayor que el especificado
SearchLess	Busca en el array ordenado el elemento menor que el especificado
SearchGreatOrEqual	Busca en el array ordenado el elemento mayor o igual que el especificado
SearchLessOrEqual	Busca en el array ordenado el elemento menor o igual que el especificado
SearchFirst	Busca en el array ordenado el primer elemento igual al especificado

© 2000-2025, MetaQuotes Ltd.

Atributos	
SearchLast	Busca en el array ordenado el último elemento igual al especificado
SearchLinear	Busca en el array el elemento igual al especificado
Entrada/salida	
virtual Save	Guarda los datos del array en el archivo
virtual Load	Carga los datos en el array a partir del archivo
virtual Type	Obtiene el identificador de tipo del array

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Delta

Establece la tolerancia de comparación.

```
void  Delta(
   double  delta      // Tolerancia
   )
```

Parámetros

delta

[in] El valor nuevo de la tolerancia de comparación.

Valor devuelto

No

Nota

Tolerancia de comparación utilizada en la búsqueda. Los valores se consideran iguales si su diferencia es menor o igual a la tolerancia. La tolerancia predeterminada es 0.0.

Ejemplo:

```
//--- ejemplo de CArrayDouble::Delta(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- establece la tolerancia de comparación
   array.Delta(0.001);
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

Asigna memoria para aumentar el tamaño del array.

```
bool  Reserve(
   int  size      // Número
   )
```

Parámetros

size

[in] Número de elementos adicionales del array.

Valor devuelto

true si se ejecuta correctamente, false - si hubo un intento de buscar una cantidad menor o igual a cero, o si el array no se incrementó.

Nota

Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayDouble::Reserve(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- reservar memoria
   if(!array.Reserve(1024))
     {
      printf("Error al reservar");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Establece un tamaño nuevo, más pequeño, del array.

```
bool  Resize(
   int  size      // Tamaño
   )
```

Parámetros

size

[in] Tamaño nuevo del array.

Valor devuelto

true si se ejecuta correctamente, false - si se intenta establecer un tamaño menor a cero.

Nota

Cambiar el tamaño del array permite un uso óptimo de la memoria. Se pierden los elementos superfluos de la derecha. Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayDouble::Resize(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- cambia el tamaño del array
   if(!array.Resize(10))
     {
      printf("Error al cambiar de tamaño");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Borra el array liberando por completo la memoria.

```
bool  Shutdown()
```

Valor devuelto

true si se ejecuta correctamente, false - si ocurre algún error.

Ejemplo:

```
//--- ejemplo de CArrayDouble::Shutdown()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- libera el array
   if(!array.Shutdown())
     {
      printf("Error al liberar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al final del array.

```
bool  Add(
   double  element      // Elemento a añadir
   )
```

Parámetros

element

[in] valor del elemento a añadir en el array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede añadir.

Ejemplo:

```
//--- ejemplo de CArrayDouble::Add(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Error al añadir el elemento");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const double&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos que se desean añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayDouble::AddArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const CArrayDouble*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase fuente CArrayDouble con los elementos fuente que se desean añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayDouble::AddArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete src;
      delete array;
      return;
     }
   //--- borra el array fuente
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta un elemento en el array en la posición especificada.

```
bool  Insert(
   double  element,     // Elemento a insertar
   int     pos          // Posición
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array

pos

[in] Posición a insertar en el array

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayDouble::Insert(double,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- insertar elementos
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Error al insertar");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array a partir de la posición especificada.

```
bool  InsertArray(
   const double&  src[],     // Array fuente
   int            pos        // Posición
   )
```

Parámetros

src[]

[in] Referencia al array con los elementos fuente a insertar

pos

[in] Posición a insertar en el array

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayDouble::InsertArray(const double &[],int)
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array a partir de la posición especificada.

```
bool  InsertArray(
   CArrayDouble*  src,      // Puntero a la fuente
   int            pos       // Posición
   )
```

Parámetros

src

[in] Puntero a la instancia fuente CArrayDouble con los elementos a insertar.

pos

[in] Posición a insertar en el array

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayDouble::InsertArray(const CArrayDouble*,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const double&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia a un array con los elementos fuente que se desean copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayDouble::AssignArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const CArrayDouble*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayDouble con los elementos fuente que se desean copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayDouble::AssignArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayDouble *src  =new CArrayDouble;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete src;
      delete array;
      return;
     }
   //--- los arrays son idénticos
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Actualiza el elemento de la posición especificada del array.

```
bool  Update(
   int     pos,          // Posición
   double  element       // Valor
   )
```

Parámetros

pos

[in] Posición en el array del elemento a cambiar

element

[in] Valor nuevo.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CArrayDouble::Update(int,double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- actualiza el elemento
   if(!array.Update(0,100.0))
     {
      printf("Error de actualización");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Mueve el ítem de una determinada posición según el desplazamiento especificado.

```
bool  Shift(
   int  pos,        // Posición
   int  shift       // Desplazamiento
   )
```

Parámetros

pos

[in] Posición del elemento desplazado

shift

[in] El valor del desplazamiento (positivo y negativo).

Valor devuelto

true si se ejecuta correctamente, false - si el ítem no se puede mover.

Ejemplo:

```
//--- ejemplo de CArrayDouble::Shift(int,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- desplaza el elemento
   if(!array.Shift(10,-5))
     {
      printf("Error de desplazamiento");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el elemento del array en la posición especificada.

```
bool  Delete(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] Posición del elemento a borrar.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede borrar.

Ejemplo:

```
//--- ejemplo de CArrayDouble::Delete(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra el elemento
   if(!array.Delete(0))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Borra un grupo de elementos de la posición especificada.

```
bool  DeleteRange(
   int  from,     // Posición del primer elemento
   int  to        // Posición del último elemento
   )
```

Parámetros

from

[in] Posición del primer elemento a partir del cual se borra el rango del array.

to

[in] Posición del último elemento que define el rango a borrar del array.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden borrar.

Ejemplo:

```
//--- ejemplo de CArrayDouble::DeleteRange(int,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra los elementos
   if(!array.DeleteRange(0,10))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del array de la posición especificada.

```
double  At(
   int  pos      // Posición
   ) const
```

Parámetros

pos

[in] Posición del elemento en el array.

Valor devuelto

El valor del elemento en caso de éxito, DBL_MAX si se intenta acceder a algún elemento en una posición que no existe (el último error ERR_OUT_OF_RANGE).

Nota

Por supuesto, DBL_MAX puede ser un valor válido del array, de modo que, dado un valor, comprobar siempre el último código de error.

Ejemplo:

```
//--- ejemplo de CArrayDouble::At(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      double result=array.At(i);
      if(result==DBL_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  Error al leer el array
         printf("Obtener el error del elemento");
         delete array;
         return;
        }
      //--- utiliza el elemento
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const double&  src[]      // Array fuente
   ) const
```

Parámetros

src[]

[in] Referencia a un array de elementos fuente para hacer la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayDouble::CompareArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const CArrayDouble*  src      // Puntero a la fuente
   ) const
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayDobule que contiene los elementos fuente para hacer la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayDouble::CompareArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra los arrays
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Minimum

Obtiene el índice del elemento menor del array en el rango especificado.

```
int  Minimum(
   int  start,      // índice de comienzo
   int  count       // número de elementos a procesar
   ) const
```

Parámetros

start

[in] Índice de comienzo del array.

count

[in] Número de elementos a procesar.

Valor devuelto

Índice del elemento menor del array en el rango especificado.

© 2000-2025, MetaQuotes Ltd.

Maximum

Obtiene el índice del elemento mayor del array en el rango especificado.

```
int  Maximum(
   int  start,      // índice de comienzo
   int  count       // número de elementos a procesar
   ) const
```

Parámetros

start

[in] Índice de comienzo del array.

count

[in] Número de elementos a procesar.

Valor devuelto

Índice del elemento mayor del array en el rango especificado.

© 2000-2025, MetaQuotes Ltd.

InsertSort

Inserta el elemento en el array ordenado.

```
bool  InsertSort(
   double  element      // Elemento a insertar
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array ordenado

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayDouble::InsertSort(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- inserta el elemento
   if(!array.InsertSort(100.0))
     {
      printf("Error de inserción");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Búsqueda

Busca un elemento igual al especificado en el array ordenado.

```
int  Search(
   double  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayDouble::Search(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.Search(100.0)!=-1) printf("Elemento encontrado");
   else                        printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Busca el elemento de más muestras en el array ordenado.

```
int  SearchGreat(
   double  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayDouble::SearchGreat(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreat(100.0)!=-1) printf("Elemento encontrado");
   else                             printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Busca un elemento menor que la muestra en el array ordenado.

```
int  SearchLess(
   double  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayDouble:: SearchLess(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLess(100.0)!=-1) printf("Elemento encontrado");
   else                            printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Busca en el array ordenado un elemento mayor o igual a la muestra.

```
int  SearchGreatOrEqual(
   double  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayDouble::SearchGreatOrEqual(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Elemento encontrado");
   else                                    printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Busca un elemento menor o igual a la muestra en el array ordenado.

```
int  SearchLessOrEqual(
   double  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayDouble::SearchLessOrEqual(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Elemento encontrado");
   else                                   printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Encuentra en el array ordenado el primer elemento igual a la muestra.

```
int  SearchFirst(
   double  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayDouble::SearchFirst(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchFirst(100.0)!=-1) printf("Elemento encontrado");
   else                             printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Encuentra el último elemento igual a la muestra en el array ordenado.

```
int  SearchLast(
   double  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayDouble::SearchLast(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLast(100.0)!=-1) printf("Elemento encontrado");
   else                            printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Busca en el array ordenado el elemento igual a la muestra.

```
int  SearchLinear(
   double  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 - si el elemento no se encuentra.

Nota

El método utiliza un algoritmo de búsqueda lineal (o secuencial) en arrays no ordenados.

Ejemplo:

```
//--- ejemplo de CArrayDouble::SearchLinear(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- busca el elemento
   if(array.SearchLinear(100.0)!=-1) printf("Elemento encontrado");
   else                              printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los datos del array en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayDouble::Save(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int            file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade 100 elementos al array
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los datos en el array a partir del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayDouble::Load(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilizar los elementos del array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = %f",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo del array.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del array (el de CArrayDouble es 87).

Ejemplo:

```
//--- ejemplo de CArrayDouble::Type()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene el tipo del array
   int type=array.Type();
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayString

La clase CArrayString proporciona un array dinámico de variables de tipo string.

Descripción

La clase CArrayString permite trabajar con un array dinámico de variables de tipo string. La clase implementa métodos para añadir, insertar y borrar elementos del array, así como ordenarlos y buscarlos. Además, los métodos implementados pueden trabajar con archivos.

Declaración

```
class CArrayString : public CArray
```

Título

```
#include <Arrays\ArrayString.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayString

Métodos de la clase

Control de memoria	
Reserve	Asigna memoria para incrementar el tamaño del array
Resize	Establece un nuevo tamaño, más pequeño, del array
Shutdown	Establece un nuevo tamaño, más pequeño, del array
Métodos de adición	
Add	Añade un elemento al final del array
AddArray	Añade al final del array los elementos de otro array
AddArray	Añade al final del array los elementos de otro array
Insert	Inserta un elemento en la posición especificada
InsertArray	Inserta en la posición especificada un array de elementos de otro array
InsertArray	Inserta en la posición especificada un array de elementos de otro array
AssignArray	Copia los elementos de otro array
AssignArray	Copia los elementos de otro array
Métodos de actualización	

© 2000-2025, MetaQuotes Ltd.

Control de memoria	
Update	Cambia el elemento de la posición especificada del array
Shift	Mueve un ítem de una posición dada a la especificada por el desplazamiento
Métodos de borrado	
Delete	Borra el elemento de la posición especificada
DeleteRange	Borra el grupo de elementos de la posición especificada
Métodos de acceso	
At	Obtiene el elemento de la posición especificada
Métodos de comparación	
CompareArray	Compara el array con otro
CompareArray	Compara el array con otro
Operaciones de ordenación de arrays	
InsertSort	Inserta el elemento en el array ordenado
Búsqueda	Busca un elemento igual a la muestra en el array ordenado
SearchGreat	Busca en el array ordenado el elemento mayor que el especificado
SearchLess	Busca en el array ordenado el elemento menor que el especificado
SearchGreatOrEqual	Busca un elemento mayor o igual a la muestra en el array ordenado
SearchLessOrEqual	Busca un elemento menor o igual a la muestra en el array ordenado
SearchFirst	Encuentra el primer elemento igual a la muestra en el array ordenado
SearchLast	Encuentra el último elemento igual a la muestra en el array ordenado
SearchLinear	Busca en el array el elemento igual al especificado
Entrada/salida	
virtual Save	Guarda los datos del array en el archivo
virtual Load	Carga los datos en el array a partir del archivo
virtual Type	Obtiene el identificador de tipo del array

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

Asigna memoria para aumentar el tamaño del array.

```
bool  Reserve(
   int  size      // Número
   )
```

Parámetros

size

[in] El número de elementos adicionales del array.

Valor devuelto

true si se ejecuta correctamente, false - si hubo un intento de buscar una cantidad menor o igual a cero, o si el array no se incrementó.

Nota

Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayString::Reserve(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- reservar memoria
   if(!array.Reserve(1024))
     {
      printf("Error al reservar");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Establece un tamaño nuevo, más pequeño, del array.

```
bool  Resize(
   int  size      // Tamaño
   )
```

Parámetros

size

[in] Tamaño nuevo del array.

Valor devuelto

true si se ejecuta correctamente, false - si se intenta establecer un tamaño menor a cero.

Nota

Cambiar el tamaño del array permite un uso óptimo de la memoria. Se pierden los elementos superfluos de la derecha. Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayString::Resize(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- cambia el tamaño del array
   if(!array.Resize(10))
     {
      printf("Error al cambiar de tamaño");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Borra el array liberando por completo la memoria.

```
bool  Shutdown()
```

Valor devuelto

true si se ejecuta correctamente, false - si ocurre algún error.

Ejemplo:

```
//--- ejemplo de CArrayString::Shutdown()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- libera el array
   if(!array.Shutdown())
     {
      printf("Error al liberar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al final del array.

```
bool  Add(
   string  element      // Elemento a añadir
   )
```

Parámetros

element

[in] valor del elemento a añadir en el array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede añadir.

Ejemplo:

```
//--- ejemplo de CArrayString::Add(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(IntegerToString(i)))
        {
         printf("Error al añadir el elemento");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const string&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia al array fuente con los elementos que se desean añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayString::AddArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const CArrayString*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayString con los elementos fuente a añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Ejemplo:

```
//--- ejemplo de CArrayString::AddArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- añade otro array
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete src;
      delete array;
      return;
     }
   //--- borra el array fuente
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta un elemento en el array en la posición especificada.

```
bool  Insert(
   string  element,     // Elemento a insertar
   int     pos          // Posición
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array

pos

[in] Posición a insertar en el array

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayString::Insert(string,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- insertar elementos
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(IntegerToString(i),0))
        {
         printf("Error al insertar");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array a partir de la posición especificada.

```
bool  InsertArray(
   const string&  src[],     // Array fuente
   int            pos        // Posición
   )
```

Parámetros

src[]

[in] Referencia al array con los elementos fuente a insertar

pos

[in] Posición a insertar en el array

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayString::InsertArray(const string &[],int)
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array a partir de la posición especificada.

```
bool  InsertArray(
   CArrayString*  src,      // Puntero a la fuente
   int             pos      // Posición
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayString con los elementos fuente a insertar.

pos

[in] Posición a insertar en el array

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Ejemplo:

```
//--- ejemplo de CArrayString::InsertArray(const CArrayString*,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- inserta otro array
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- borra el array fuente
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const string&  src[]      // Array fuente
   )
```

Parámetros

src[]

[in] Referencia a un array con los elementos fuente que se desean copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayString::AssignArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const CArrayString*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayString con los elementos fuente a copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Ejemplo:

```
//--- ejemplo de CArrayString::AssignArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayString *src  =new CArrayString;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- asignar otro array
   if(!array.AssignArray(src))
     {
      printf("Error en la asignación del array");
      delete src;
      delete array;
      return;
     }
   //--- los arrays son idénticos
   //--- borra el array fuente
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Actualiza el elemento de la posición especificada del array.

```
bool  Update(
   int     pos,          // Posición
   string  element       // Valor
   )
```

Parámetros

pos

[in] Posición en el array del elemento a cambiar

element

[in] Valor nuevo del elemento

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CArrayString::Update(int, string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- actualiza el elemento
   if(!array.Update(0,"ABC"))
     {
      printf("Error de actualización");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Mueve el ítem de una determinada posición según el desplazamiento especificado.

```
bool  Shift(
   int  pos,        // Posición
   int  shift       // Desplazamiento
   )
```

Parámetros

pos

[in] Posición del elemento desplazado

shift

[in] El valor del desplazamiento (positivo y negativo).

Valor devuelto

true si se ejecuta correctamente, false - si el ítem no se puede mover.

Ejemplo:

```
//--- ejemplo de CArrayString::Shift(int,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- desplaza el elemento
   if(!array.Shift(10,-5))
     {
      printf("Error de desplazamiento");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el elemento del array en la posición especificada.

```
bool  Delete(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] Posición del elemento a borrar.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede borrar.

Ejemplo:

```
//--- ejemplo de CArrayString::Delete(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra el elemento
   if(!array.Delete(0))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Borra un grupo de elementos de la posición especificada.

```
bool  DeleteRange(
   int  from,     // Posición del primer elemento
   int  to        // Posición del último elemento
   )
```

Parámetros

from

[in] Posición del primer elemento a partir del cual se borra el rango del array.

to

[in] Posición del último elemento que define el rango a borrar del array.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden borrar.

Ejemplo:

```
//--- ejemplo de CArrayString::DeleteRange(int,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra los elementos
   if(!array.DeleteRange(0,10))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del array de la posición especificada.

```
string  At(
   int  pos      // Posición
   ) const
```

Parámetros

pos

[in] Posición del elemento en el array.

Valor devuelto

El valor del elemento, si se ejecuta correctamente; si se intenta acceder a un elemento de una posición que no existe (el último error ERR_OUT_OF_RANGE).

Nota

Por supuesto, puede tratarse de un valor válido de un elemento del array, de modo que, dado un valor, comprobar siempre el último código de error.

Ejemplo:

```
//--- ejemplo de CArrayString::At(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      string result=array.At(i);
      if(result=="" && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- Error al leer el array
         printf("Obtener el error del elemento");
         delete array;
         return;
        }
      //--- utiliza el elemento
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const string&  src[]      // Array fuente
   ) const
```

Parámetros

src[]

[in] Referencia a un array de elementos fuente para hacer la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayString::CompareArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArrays(
   const CArrayString*  src      // Puntero a la fuente
   ) const
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayString que contiene los elementos fuente para hacer la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayString::CompareArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- añade elementos
   //--- . . .
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra los arrays
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Inserta el elemento en el array ordenado.

```
bool  InsertSort(
   string  element      // Elemento a insertar
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array ordenado

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Ejemplo:

```
//--- ejemplo de CArrayString::InsertSort(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- inserta el elemento
   if(!array.InsertSort("ABC"))
     {
      printf("Error de inserción");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Búsqueda

Busca un elemento igual al especificado en el array ordenado.

```
int  Search(
   string  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayString::Search(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.Search("ABC")!=-1) printf("Elemento encontrado");
   else                        printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Busca el elemento de más muestras en el array ordenado.

```
int  SearchGreat(
   string  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayString::SearchGreat(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreat("ABC")!=-1) printf("Elemento encontrado");
   else                             printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Busca un elemento menor que la muestra en el array ordenado.

```
int  SearchLess(
   string  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayString:: SearchLess(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLess("ABC")!=-1) printf("Elemento encontrado");
   else                            printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Busca en el array ordenado un elemento mayor o igual a la muestra.

```
int  SearchGreatOrEqual(
   string  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayString:: SearchGreatOrEqual(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchGreatOrEqual("ABC")!=-1) printf("Elemento encontrado");
   else                                    printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Busca un elemento menor o igual a la muestra en el array ordenado.

```
int  SearchLessOrEqual(
   string  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayString:: SearchLessOrEqual(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLessOrEqual("ABC")!=-1) printf("Elemento encontrado");
   else                                   printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Encuentra en el array ordenado el primer elemento igual a la muestra.

```
int  SearchFirst(
   string  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayString:: SearchFirst(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchFirst("ABC")!=-1) printf("Elemento encontrado");
   else                             printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Encuentra el último elemento igual a la muestra en el array ordenado.

```
int  SearchLast(
   string  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayString:: SearchLast(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- busca el elemento
   if(array.SearchLast("ABC")!=-1) printf("Elemento encontrado");
   else                            printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Busca el elemento igual a la muestra en el array.

```
int  SearchLinear(
   string  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 - si el elemento no se encuentra.

Nota

El método utiliza un algoritmo de búsqueda lineal (o secuencial) en arrays no ordenados.

Ejemplo:

```
//--- ejemplo de CArrayString::SearchLinear(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- busca el elemento
   if(array.SearchLinear("ABC")!=-1) printf("Elemento encontrado");
   else                              printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los datos del array en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayString::Save(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int            file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade 100 elementos al array
   for(int i=0;i<100;i++)
     {
      array.Add(IntegerToString(i));
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los datos en el array a partir del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayString::Load(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilizar los elementos del array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = '%s'",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo del array.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de array (el de CArrayString es 89).

Ejemplo:

```
//--- ejemplo de CArrayString::Type()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene el tipo del array
   int type=array.Type();
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayObj

La clase CArrayObj proporciona un array dinámico de punteros a instancias de la clase CObject y a sus clases hijas.

Descripción

La clase CArrayObj permite trabajar con un array dinámico de punteros a instancias de CObject y a sus clases hijas. Esto brinda la posibilidad de trabajar con arrays dinámicos multidimensionales de tipos de datos primitivos, y realizar cosas más complejas con estructuras de datos organizadas.

Esta clase permite añadir, insertar y borrar elementos del array, así como ordenarlos y buscarlos en arrays ordenados. Además, los métodos implementados permiten trabajar con archivos.

La clase CArrayObj tiene algunas sutilezas.

Declaración

```
class CArrayObj : public CArray
```

Título

```
#include <Arrays\ArrayObj.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj

Descendientes directos

CIndicators, CSeries

Método de la clase

Atributos	
FreeMode	Obtiene la bandera de gestión de memoria
FreeMode	Establece la bandera de gestión de memoria
Control de memoria	
Reserve	Asigna memoria para incrementar el tamaño del array
Resize	Establece un nuevo tamaño, más pequeño, del array
Shutdown	Borra el array con un array totalmente exento de memoria (sin elementos).
Métodos de adición	
Add	Añade un elemento al final del array

© 2000-2025, MetaQuotes Ltd.

Atributos	
AddArray	Añade un elemento al final del array
Insert	Inserta un elemento en la posición especificada
InsertArray	Inserta en la posición especificada un array de elementos de otro array
AssignArray	Copia los elementos de otro array
Métodos de actualización	
Update	Cambia el elemento de la posición especificada del array
Shift	Mueve un ítem de una posición dada a la especificada por el desplazamiento
Métodos de borrado	
Detach	Obtiene el elemento de la posición especificada y lo borra del array
Delete	Borra el elemento de la posición especificada
DeleteRange	Borra el grupo de elementos de la posición especificada
Clear	Borra todos los elementos del array sin liberar la memoria
Métodos de acceso	
At	Obtiene el elemento de la posición especificada
Métodos de comparación	
CompareArray	Compara el array con otro
Operaciones de ordenación	
InsertSort	Inserta el elemento en el array ordenado
Búsqueda	Busca en el array ordenado un elemento igual al especificado
SearchGreat	Busca en el array ordenado el elemento mayor que el especificado
SearchLess	Busca en el array ordenado el elemento menor que el especificado
SearchGreatOrEqual	Busca en el array ordenado el elemento mayor o igual que el especificado
SearchLessOrEqual	Busca en el array ordenado el elemento menor o igual que el especificado
SearchFirst	Busca en el array ordenado el primer elemento igual al especificado
SearchLast	Busca en el array ordenado el último elemento igual al especificado
Entrada/salida	
Save	Guarda los datos del array en el archivo

© 2000-2025, MetaQuotes Ltd.

Atributos	
Load	Carga los datos en el array a partir del archivo
Type	Obtiene el identificador de tipo del array

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Las aplicaciones prácticas de los arrays descienden de la clase CObject (incluyendo todas las clases de la Librería Estándar).

Por ejemplo, consideremos un array de dos dimensiones:

```
#include <Arrays\ArrayDouble.mqh>
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int i,j;
   int first_size=10;
   int second_size=100;
//--- crea el array
   CArrayObj    *array=new CArrayObj;
   CArrayDouble *sub_array;
//---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
//--- crea subarrays
   for(i=0;i<first_size;i++)
     {
      sub_array=new CArrayDouble;
      if(sub_array==NULL)
        {
         delete array;
         printf("Error en la creación del objeto");
         return;
        }
      //--- llena el array
      for(j=0;j<second_size;j++)
        {
         sub_array.Add(i*j);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      array.Add(sub_array);
     }
//--- el array se crea correctamente
   for(i=0;i<first_size;i++)
     {
      sub_array=array.At(i);
      for(j=0;j<second_size;j++)
        {
         double element=sub_array.At(j);
         //--- utiliza el elemento del array
        }
     }
   delete array;
  }
```

Sutilezas

La clase tiene un mecanismo de control de memoria dinámica, de modo que tenga cuidado cuando tenga que manejar los elementos del array.

El mecanismo de gestión de la memoria se puede activar y desactivar utilizando el método FreeMode (bool). El mecanismo está activado de forma predeterminada.

En consecuencia, hay dos opciones para trabajar con la clase CArrayObj:

1. Activar el mecanismo de gestión de la memoria. (predeterminado)

En este caso, CArrayObj se responsabiliza de liberar los elementos de la memoria después de eliminarse del array. En el siguiente programa, el usuario no debe liberar los elementos del array.

Ejemplo:

```
   int i;
//--- Crea un array
   CArrayObj *array=new CArrayObj;
//--- Inserta los elementos en el array
   for(i=0;i<10;i++) array.Add(new CObject);
//--- Hace algo
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- Acción con un elemento
      . . .
     }
//--- Borrar el array con los elementos
   delete array;
```

2. El mecanismo de gestión de memoria está desactivado.

En este caso, CArrayObj se responsabiliza de liberar la memoria utilizada por los elementos después de borrarlos del array. En este programa, el usuario debe liberar los elementos del array.

© 2000-2025, MetaQuotes Ltd.

Ejemplo:

```
   int i;
//--- Crea un array
   CArrayObj *array=new CArrayObj;
//--- Desactiva el mecanismo de gestión de memoria
   array.FreeMode(false);
//--- Inserta los elementos en el array
   for(i=0;i<10;i++) array.Add(new CObject);
//--- Hace algo
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- Acción con un elemento
      . . .
     }
//--- Borra los elementos del array
   while(array.Total()) delete array.Detach();
//--- Borra el array vacío
   delete array;
```

© 2000-2025, MetaQuotes Ltd.

FreeMode

Obtiene la bandera de gestión de memoria.

```
bool  FreeMode() const
```

Valor devuelto

Bandera de gestión de memoria.

Ejemplo:

```
//--- ejemplo de CArrayObj::FreeMode()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene la bandera
   bool array_free_mode=array.FreeMode();
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

FreeMode

Establece la bandera de gestión de memoria.

```
void  FreeMode(
   bool  mode      // Bandera nueva
   )
```

Parámetros

mode

[in] Valor nuevo de la bandera de gestión de memoria.

Valor devuelto

Ninguno.

Nota

Configurar la bandera de gestión de memoria es un aspecto importante en el uso de la clase CArrayObj. Dado que los elementos del array son punteros a objetos dinámicos es importante determinar qué hacer con ellos al borrarlos del array.

Si se establece la bandera, al eliminar un elemento del array, el elemento se borra automáticamente por medio del operador delete. Si no se establece la bandera, se asume que todavía existe un puntero al objeto eliminado en algún lugar del programa, y el programa lo liberará a continuación.

Si el usuario reinicia la bandera de gestión de memoria, entonces debe entender que tiene que responsabilizarse de la eliminación del array antes de que finalice el programa, porque de lo contrario, la memoria será ocupada por otros elementos al crear el nuevo operador.

Si el usuario no reinicia la bandera de gestión de memoria, se puede incluso estropear el terminal al manejar grandes cantidades de datos.

Almacenar punteros y arrays en variables locales después de borrar el array, dará lugar a errores críticos y bloqueos del programa. De forma predeterminada, queda establecida la bandera de gestión de memoria; la clase del array se responsabiliza de liberar la memoria de los elementos.

Ejemplo:

```
//--- ejemplo de CArrayObj::FreeMode(bool)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- reiniciar la bandera
   array.FreeMode(false);
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

Asigna memoria para aumentar el tamaño del array.

```
bool  Reserve(
   int  size      // Número
   )
```

Parámetros

size

[in] El número de elementos adicionales del array.

Valor devuelto

true si se ejecuta correctamente, false - si hubo un intento de buscar una cantidad menor o igual a cero, o si el array no se incrementó.

Nota

Para reducir la fragmentación de la memoria, el incremento del tamaño del array se lleva a cabo en un paso previo mediante el método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayObj::Reserve(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   if(!array.Reserve(1024))
     {
      printf("Error al reservar");
      delete array;
      return;
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Establece un tamaño nuevo, más pequeño, del array.

```
bool  Resize(
   int  size      // Tamaño
   )
```

Parámetros

size

[in] Tamaño nuevo del array.

Valor devuelto

true si se ejecuta correctamente, false - si se intenta establecer un tamaño menor a cero.

Nota

Cambiar el tamaño del array permite utilizar la memoria de manera óptima. Se pierden los elementos situados a la derecha. La memoria de los elementos perdidos se puede liberar o no, dependiendo del modo de gestión de memoria.

Para reducir la fragmentación de la memoria, el cambio de tamaño del array se lleva a cabo en un paso previo por medio del método Step (int), o 16 (predeterminado).

Ejemplo:

```
//--- ejemplo de CArrayObj::Resize(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- cambia el tamaño del array
   if(!array.Resize(10))
     {
      printf("Error al cambiar de tamaño");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

Borra todos los elementos del array sin liberar la memoria.

```
void  Clear()
```

Valor devuelto

No.

Nota

Si la gestión de memoria está activada, la memoria utilizada en los elementos borrados se libera.

Ejemplo:

```
//--- ejemplo de CArrayObj::Clear()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra el array
   array.Clear();
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Borra el array con un array totalmente exento de memoria (sin elementos).

```
bool  Shutdown()
```

Valor devuelto

true si se ejecuta correctamente, false - si ocurre algún error.

Nota

Si la gestión de memoria está activada, la memoria utilizada en los elementos borrados se libera.

Ejemplo:

```
//--- ejemplo de CArrayObj::Shutdown()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- libera el array
   if(!array.Shutdown())
     {
      printf("Error al liberar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateElement

Crea un nuevo elemento en el array en la posición especificada.

```
bool  CreateElement(
   int  index      // Posición
   )
```

Parámetros

index

[in] Posición donde se desea crear un elemento nuevo.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede crear.

Nota

El método CreateElement (int) de la clase CArrayObj siempre devuelve false y no lleva a cabo ninguna acción. Si es necesario, implementar el método CreateElement (int) en una clase hija.

Ejemplo:

```
//--- ejemplo de CArrayObj::CreateElement(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int        size=100;
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- llenar array
   array.Reserve(size);
   for(int i=0;i<size;i++)
     {
      if(!array.CreateElement(i))
        {
         printf("Error en la creación del elemento");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al final del array.

```
bool  Add(
   CObject*  element      // Elemento a añadir
   )
```

Parámetros

element

[in] valor del elemento a añadir en el array.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede añadir.

Nota

El elemento no se añade al array si el valor no es un puntero válido (tal como NULL).

Ejemplo:

```
//--- ejemplo de CArrayObj::Add(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade 100 elementos al array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(new CObject))
        {
         printf("Error al añadir el elemento");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Añade al final del array los elementos de otro array.

```
bool  AddArray(
   const CArrayObj *  src      // Puntero al array fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayDouble que contiene los elementos fuente que se desean añadir.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden añadir.

Nota

En realidad, la adición de elementos en el array consiste en una copia de punteros. Por lo tanto, al llamar a este método, hay una referencia que puede ser un puntero a un objeto dinámico de más de una variable.

```
//--- ejemplo
extern bool       make_error;
extern int        error;
extern CArrayObj *src;
//--- Crear una nueva instancia de CArrayObj
//--- La gestión predeterminada de la memoria está activada
CArrayObj *array=new CArrayObj;
//--- Añadir (copiar) los elementos del array fuente
if(array!=NULL)
   bool result=array.AddArray(src);
if(make_error)
  {
   //--- Realizar las acciones de los errores
   switch(error)
     {
      case 0:
         //--- Borrar el array fuente, sin comprobar su bandera de gestión de memoria
         delete src;
         //--- Resultado:
         //--- Se puede direccionar un elemento con un puntero no válido del array rec
         break;
      case 1:
         //--- Desactiva el mecanismo de gestión de memoria del array fuente
         if(src.FreeMode()) src.FreeMode(false);
         //--- Sin eliminar el array fuente
         //--- Resultado:
         //--- Después de borrar el array receptor se puede direccionar un elemento co
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
      case 2:
         //--- Desactiva el mecanismo de gestión de memoria del array fuente
         src.FreeMode(false);
         //--- Desactiva el mecanismo de gestión de memoria del array receptor
         array.FreeMode(false);
         //--- Resultado:
         //--- Después de terminar el programa se obtiene una "pérdida de memoria"
         break;
      }
  }
else
  {
   //--- Desactiva el mecanismo de gestión de memoria del array fuente
   if(src.FreeMode()) src.FreeMode(false);
   //--- Borra el array fuente
   delete src;
   //--- Resultado:
   //--- Direccionar el elemento del array receptor será correcto
   //--- Borrar el array receptor implicará borrar sus elementos
  }
```

Ejemplo:

```
//--- ejemplo de CArrayObj::AddArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- reiniciar la bandera
   src.FreeMode(false);
   //--- llenar el array fuente
   //--- . . .
   //--- añade otro array
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!array.AddArray(src))
     {
      printf("Error al añadir el array");
      delete src;
      delete array;
      return;
     }
   //--- borrar el array fuente sin elementos
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta un elemento en el array en la posición especificada.

```
bool  Insert(
   CObject*  element,     // Elemento a insertar
   int       pos          // Posición
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array

pos

[in] Posición a insertar en el array

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Nota

El elemento no se añade al array si el valor no es un puntero válido (tal como NULL).

Ejemplo:

```
//--- ejemplo de CArrayObj::Insert(CObject*,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- insertar elementos
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(new CObject,0))
        {
         printf("Error al insertar");
         delete array;
         return;
        }
     }
   //--- utiliza el array
   //--- . . .
   //--- borra el array
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Inserta un array de elementos de otro array a partir de la posición especificada.

```
bool  InsertArray(
   const CArrayObj*  src,     // Puntero a la fuente
   int               pos      // Posición
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayObj con los elementos fuente a insertar.

pos

[in] Posición a insertar en el array

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden insertar.

Nota

Ver: CArrayObj::AddArray(const CArrayObj*).

Ejemplo:

```
//--- ejemplo de CArrayObj::InsertArray(const CArrayObj*,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- reiniciar la bandera
   src.FreeMode(false);
   //--- llenar el array fuente
   //--- . . .
   //--- inserta otro array
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!array.InsertArray(src,0))
     {
      printf("Error de inserción en el array");
      delete src;
      delete array;
      return;
     }
   //--- borrar el array fuente sin elementos
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Copia los elementos de otro array.

```
bool  AssignArray(
   const CArrayObj*  src      // Puntero a la fuente
   )
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayObj - elementos fuente a copiar.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden copiar.

Nota

Si el array receptor AssignArray no está vacío, todos sus elementos serán eliminados; y si la bandera de gestión de memoria está activada, la memoria utilizada en los ítemes borrados se libera. El array receptor es una copia exacta del array fuente. Para más información consultar CArrayObj::AddArray(const CArrayObj*).

Ejemplo:

```
//--- ejemplo de CArrayObj::AssignArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- reiniciar la bandera
   src.FreeMode(false);
   //--- llenar el array fuente
   //--- . . .
   //--- asignar otro array
   if(!array.AssignArray(src))
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      printf("Error en la asignación del array");
      delete src;
      delete array;
      return;
     }
   //--- los arrays son idénticos
   //--- borrar el array fuente sin elementos
   delete src;
   //--- utiliza el array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Actualiza el elemento de la posición especificada del array.

```
bool  Update(
   int       pos,          // Posición
   CObject*  element       // Valor
   )
```

Parámetros

pos

[in] Posición en el array del elemento a cambiar

element

[in] Valor nuevo del elemento

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede cambiar.

Nota

El elemento no cambia si se le pasa como parámetro un puntero no válido, por ejemplo NULL. Si se habilita la gestión de memoria, se libera el marcador de posición de memoria.

Ejemplo:

```
//--- ejemplo de CArrayObj::Update(int,CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- actualiza el elemento
   if(!array.Update(0,new CObject))
     {
      printf("Error de actualización");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Mueve el ítem de una determinada posición según el desplazamiento especificado.

```
bool  Shift(
   int  pos,        // Posición
   int  shift       // Desplazamiento
   )
```

Parámetros

pos

[in] Posición del elemento desplazado

shift

[in] El valor del desplazamiento (positivo y negativo).

Valor devuelto

true si se ejecuta correctamente, false - si el ítem no se puede mover.

Ejemplo:

```
//--- ejemplo de CArrayObj::Shift(int,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- desplaza el elemento
   if(!array.Shift(10,-5))
     {
      printf("Error de desplazamiento");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Detach

Borra un elemento del array en la posición especificada.

```
CObject*  Detach(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] Posición del elemento en el array.

Valor devuelto

Puntero a la eliminación de los elementos en caso de éxito, NULL - si no se puede eliminar el elemento.

Nota

Cuando se elimina del array, el elemento no se borra en ningún estado de la bandera de gestión de memoria. Puntero al elemento retirado.

Ejemplo:

```
//--- ejemplo de CArrayObj::Detach(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   CObject *object=array.Detach(0);
   if(object==NULL)
     {
      printf("Quitar error");
      delete array;
      return;
     }
   //--- utiliza el elemento
   //--- . . .
   //--- borra el elemento
   delete object;
   //--- borra el array
   delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el elemento del array en la posición especificada.

```
bool  Delete(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] Posición del elemento a borrar.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede borrar.

Nota

Si la gestión de memoria está activada, la memoria utilizada en los elementos borrados se libera.

Ejemplo:

```
//--- ejemplo de CArrayObj::Delete(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   if(!array.Delete(0))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Borra un grupo de elementos de la posición especificada.

```
bool  DeleteRange(
   int  from,     // Posición del primer elemento
   int  to        // Posición del último elemento
   )
```

Parámetros

from

[in] Posición del primer elemento a partir del cual se borra el rango del array.

to

[in] Posición del último elemento que define el rango a borrar del array.

Valor devuelto

true si se ejecuta correctamente, false - si los elementos no se pueden borrar.

Nota

Si la gestión de memoria está activada, la memoria utilizada en los elementos borrados se libera.

Ejemplo:

```
//--- ejemplo de CArrayObj::DeleteRange(int,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- borra los elementos
   if(!array.DeleteRange(0,10))
     {
      printf("Error al borrar");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del array de la posición especificada.

```
CObject*  At(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] Posición del elemento en el array.

Valor devuelto

El valor del elemento, si se ejecuta correctamente; NULL si se intenta acceder a un elemento en una posición que no existe.

Ejemplo:

```
//--- ejemplo de CArrayObj::At(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      CObject *result=array.At(i);
      if(result==NULL)
        {
         //--- Error al leer el array
         printf("Obtener el error del elemento");
         delete array;
         return;
        }
      //--- utiliza el elemento
      //--- . . .
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Compara el array con otro array.

```
bool  CompareArray(
   const CArrayObj*  src      // Puntero a la fuente
   ) const
```

Parámetros

src

[in] Puntero a una instancia de la clase CArrayObj que contiene los elementos fuente para realizar la comparación.

Valor devuelto

true si los arrays son iguales, false - si no lo son.

Ejemplo:

```
//--- ejemplo de CArrayObj::CompareArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- crea el array fuente
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Error en la creación del objeto");
      delete array;
      return;
     }
   //--- llenar el array fuente
   //--- . . .
   //--- compara con otro array
   int result=array.CompareArray(src);
   //--- borra los arrays
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Inserta el elemento en el array ordenado.

```
bool  InsertSort(
   CObject*  element      // Elemento a insertar
   )
```

Parámetros

element

[in] Valor del elemento a insertar en el array ordenado

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede insertar.

Nota

El elemento no se añade al array si el valor no es un puntero válido (tal como NULL).

Ejemplo:

```
//--- ejemplo de CArrayObj::InsertSort(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- inserta el elemento
   if(!array.InsertSort(new CObject))
     {
      printf("Error de inserción");
      delete array;
      return;
     }
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Búsqueda

Busca un elemento igual al especificado en el array ordenado.

```
int  Search(
   CObject*  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayObj::Search(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- crea la muestra
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Error al crear la muestra");
      delete array;
      return;
     }
   //--- establece los atributos de la muestra
   //--- . . .
   //--- busca el elemento
   if(array.Search(sample)!=-1) printf("Elemento encontrado");
   else                         printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Busca el elemento de más muestras en el array ordenado.

```
int  SearchGreat(
   CObject*  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayObj::SearchGreat(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- crea la muestra
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Error al crear la muestra");
      delete array;
      return;
     }
   //--- establece los atributos de la muestra
   //--- . . .
   //--- busca el elemento
   if(array.SearchGreat(sample)!=-1) printf("Elemento encontrado");
   else                              printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Busca un elemento menor que la muestra en el array ordenado.

```
int  SearchLess(
   CObject*  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayObj:: SearchLess(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- crea la muestra
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Error al crear la muestra");
      delete array;
      return;
     }
   //--- establece los atributos de la muestra
   //--- . . .
   //--- busca el elemento
   if(array.SearchLess(sample)!=-1) printf("Elemento encontrado");
   else                             printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Busca en el array ordenado un elemento mayor o igual a la muestra.

```
int  SearchGreatOrEqual(
   CObject*  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayObj::SearchGreatOrEqual(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- crea la muestra
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Error al crear la muestra");
      delete array;
      return;
     }
   //--- establece los atributos de la muestra
   //--- . . .
   //--- busca el elemento
   if(array.SearchGreatOrEqual(sample)!=-1) printf("Elemento encontrado");
   else                                     printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Busca un elemento menor o igual a la muestra en el array ordenado.

```
int  SearchLessOrEqual(
   CObject*  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayObj:: SearchLessOrEqual(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- crea la muestra
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Error al crear la muestra");
      delete array;
      return;
     }
   //--- establece los atributos de la muestra
   //--- . . .
   //--- busca el elemento
   if(array.SearchLessOrEqual(sample)!=-1) printf("Elemento encontrado");
   else                                    printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Encuentra en el array ordenado el primer elemento igual a la muestra.

```
int  SearchFirst(
   CObject*  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayObj::SearchFirst(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- crea la muestra
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Error al crear la muestra");
      delete array;
      return;
     }
   //--- establece los atributos de la muestra
   //--- . . .
   //--- busca el elemento
   if(array.SearchFirst(sample)!=-1) printf("Elemento encontrado");
   else                              printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Encuentra el último elemento igual a la muestra en el array ordenado.

```
int  SearchLast(
   CObject*  element      // Muestra
   ) const
```

Parámetros

element

[in] El elemento muestra a buscar en el array.

Valor devuelto

La posición del elemento encontrado, si se ejecuta correctamente, -1 si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CArrayObj:: SearchLast(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- ordena el array
   array.Sort();
   //--- crea la muestra
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Error al crear la muestra");
      delete array;
      return;
     }
   //--- establece los atributos de la muestra
   //--- . . .
   //--- busca el elemento
   if(array.SearchLast(sample)!=-1) printf("Elemento encontrado");
   else                             printf("Elemento no encontrado");
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los datos del array en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador de archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CArrayObj::Save(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos al array
   //--- . . .
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los datos del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Nota

Cuando se lee el archivo para crear los elementos del array se llama al método CArrayObj::CreateElement(int).

Ejemplo:

```
//--- ejemplo de CArrayObj::Load(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- utilizar los elementos del array
   //--- . . .
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo del array.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del array (el de CArrayObj es 7778).

Ejemplo:

```
//--- ejemplo de CArrayObj::Type()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene el tipo del array
   int type=array.Type();
   //--- borra el array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CList

La clase CList proporciona una lista dinámica de instancias de la clase CObject e instancias de clases hijas.

Descripción

La clase CList permite trabajar con una lista de instancias de CObject e instancias de clases hijas. La clase permite añadir, insertar y borrar elementos de la lista, así como ordenarlos y buscarlos en una lista ordenada. Además, los métodos implementados permiten trabajar con archivos.

La clase CList presenta algunas sutilezas. Esta clase tiene un mecanismo de control de memoria dinámica, de modo que tenga cuidado al trabajar con los elementos de la lista.

Las sutilezas del mecanismo de gestión de memoria son parecidas a las descritas en CArrayObj.

Declaración

```
class CList : public CObject
```

Título

```
#include <Arrays\List.mqh>
```

Jerarquía de herencia

CObject
  CList

Métodos de la clase

Atributos	
FreeMode	Obtiene la bandera de gestión de memoria al borrar elementos de la lista.
FreeMode	Establece la bandera de gestión de memoria al borrar elementos de la lista
Total	Obtiene el número de elementos de la lista
IsSorted	Obtiene la bandera de la lista ordenada
SortMode	Obtiene el modo de ordenación
Métodos de creación	
CreateElement	Crea un nuevo ítem en la lista
Métodos de adición	
Add	Añade un elemento al final de la lista
Insert	Inserta un elemento en la lista en la posición especificada

© 2000-2025, MetaQuotes Ltd.

Atributos	
Métodos de borrado	
DetachCurrent	Borra el elemento de la lista de la posición actual sin eliminarlo "físicamente"
DeleteCurrent	Borra el elemento de la lista de la posición actual
Delete	Borra el elemento de la posición especificada de la lista
Clear	Borra todos los ítemes
Navegación	
IndexOf	Obtiene el índice del ítem de la lista
GetNodeAtIndex	Obtiene el ítem de la lista del índice especificado
GetFirstNode	Obtiene el primer elemento de la lista
GetPrevNode	Obtiene el elemento anterior de la lista
GetCurrentNode	Obtiene el elemento actual de la lista
GetNextNode	Obtiene el próximo elemento de la lista
GetLastNode	Obtiene el último elemento
Métodos de ordenación	
Sort	Ordena la lista
MoveToIndex	Mueve el elemento actual de la lista a la posición especificada
Exchange	Intercambia los elementos de la lista
Métodos de comparación	
CompareList	Compara la lista con otra
Métodos de búsqueda	
Búsqueda	Busca un elemento igual a la muestra en la lista ordenada
Entrada/salida	
virtual Save	Guarda los datos en el archivo
virtual Load	Carga los datos del archivo
virtual Type	Obtiene el identificador de tipo de la lista

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

© 2000-2025, MetaQuotes Ltd.

FreeMode

Obtiene la bandera de gestión de memoria al borrar elementos de la lista.

```
bool  FreeMode() const
```

Valor devuelto

Bandera de gestión de memoria.

Ejemplo:

```
//--- ejemplo de CList::FreeMode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- obtiene la bandera
   bool list_free_mode=list.FreeMode();
   //--- borra la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

FreeMode

Establece la bandera de gestión de memoria al borrar elementos de la lista.

```
void  FreeMode(
   bool  mode      // Valor nuevo
   )
```

Parámetros

mode

[in] Valor nuevo de la bandera de gestión de memoria.

Nota

Configurar la bandera de gestión de memoria es un aspecto importante en el uso de la clase CList. Dado que los elementos de la lista son punteros a objetos dinámicos es importante determinar qué hacer con ellos al borrarlos de la lista. Si se establece la bandera, al eliminar un elemento de la lista, el elemento se borra automáticamente por medio del operador delete. Si no se establece la bandera, se asume que todavía existe un puntero al objeto eliminado en algún lugar del programa, y el programa lo liberará a continuación.

Si el usuario reinicia la bandera de gestión de memoria, entonces debe entender que tiene que responsabilizarse de la eliminación de los elementos de la lista antes de que finalice el programa, porque de lo contrario, la memoria será ocupada por otros elementos al crear el nuevo operador. Si el usuario no reinicia la bandera de gestión de memoria, se puede incluso estropear el terminal al manejar grandes cantidades de datos.

Almacenar punteros a listas en variables locales después de borrar la lista, dará lugar a errores críticos y bloqueos del programa. De forma predeterminada, queda establecida la bandera de gestión de memoria; la clase de la lista se responsabiliza de liberar la memoria de los elementos.

Ejemplo:

```
//--- ejemplo de CList::FreeMode(bool)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- reinicia la bandera
   list.FreeMode(false);
   //--- utiliza la lista
   //--- . . .
   //--- borra la lista
   delete list;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Total

Obtiene el número de elementos de la lista.

```
int  Total() const
```

Valor devuelto

Número de elementos de la lista.

Ejemplo:

```
//--- ejemplo de CList::Total()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- comprueba el total
   int total=list.Total();
   //--- utiliza la lista
   //--- ...
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

IsSorted

Obtiene la bandera de ordenación.

```
bool  IsSorted(
   int  mode=0      // Modo de ordenación
   ) const
```

Parámetros

mode=0

[in] Versión de la ordenación

Valor devuelto

Bandera de la lista ordenada. Devuelve true si la lista está ordenada de acuerdo al modo especificado. false; en caso contrario. false.

Nota

La bandera de la lista ordenada no se puede cambiar directamente. Bandera establecida por Sort (int) y reinicia los métodos para añadir e insertar.

Ejemplo:

```
//--- ejemplo de CList::IsSorted()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- comprueba la ordenación
   if(list.IsSorted(0))
     {
      //--- utiliza los métodos de la lista ordenada
      //--- ...
     }
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

SortMode

Obtiene el modo de ordenación.

```
int  SortMode() const
```

Valor devuelto

Modo de ordenación, o -1 si la lista no está ordenada.

Ejemplo:

```
//--- ejemplo de CList::SortMode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- comprueba el modo de ordenación
   int sort_mode=list.SortMode();
   //--- utiliza la lista
   //--- ...
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateElement

Crea un nuevo ítem en la lista.

```
CObject*  CreateElement()
```

Valor devuelto

Puntero al elemento recién creado, si se ejecuta correctamente; NULL, si el elemento no se puede crear.

Nota

El método CreateElement () de la clase CList siempre devuelve NULL y no lleva a cabo ninguna acción. Si es necesario, se puede implementar el método CreateElement () en una clase derivada.

Ejemplo:

```
//--- ejemplo de CList::CreateElement(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    size=100;
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- llena la lista
   for(int i=0;i<size;i++)
     {
      CObject *object=list.CreateElement();
      if(object==NULL)
        {
         printf("Error al crear el elemento");
         delete list;
         return;
        }
      list.Add(object);
     }
   //--- utiliza la lista
   //--- . . .
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al final de la lista.

```
int  Add(
   CObject*  element      // Elemento a añadir
   )
```

Parámetros

element

[in] Valor del elemento a añadir en la lista.

Valor devuelto

Si se ejecuta correctamente, devuelve el índice del elemento añadido; en caso de error devuelve -1.

Nota

Si el parámetro no contiene un puntero válido (por ejemplo, NULL), entonces el elemento no se añade a la lista.

Ejemplo:

```
//--- ejemplo de CList::Add(Cobject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade 100 elementos
   for(int i=0;i<100;i++)
     {
      if(list.Add(new CObject)==-1)
        {
         printf("Error al añadir elemento");
         delete list;
         return;
        }
     }
   //--- utiliza la lista
   //--- . . .
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta el elemento en la lista en la posición especificada.

```
int  Insert(
   CObject*  element,     // Elemento a insertar
   int       pos          // Posición
   )
```

Parámetros

element

[in] valor del elemento a insertar en la lista

pos

[in] Posición a insertar en la lista

Valor devuelto

Devuelve el índice del elemento insertado en caso de ejecutarse correctamente; o -1 en caso de error.

Nota

Si el parámetro no contiene un puntero válido (por ejemplo, NULL), entonces el elemento no se añade a la lista.

Ejemplo:

```
//--- ejemplo de CList::Insert(CObject*,int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- inserta 100 elementos
   for(int i=0;i<100;i++)
     {
      if(list.Insert(new CObject,0)==-1)
        {
         printf("Error al insertar elemento");
         delete list;
         return;
        }
     }
   //--- utiliza la lista
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- . . .
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

DetachCurrent

Extrae un elemento de la posición actual sin borrarlo de forma física.

```
CObject*  DetachCurrent()
```

Valor devuelto

Puntero a la eliminación de los elementos en caso de éxito, NULL - si no se puede eliminar el elemento.

Nota

Cuando se elimina de la lista, el elemento no se borra en ninguno de los estados de la bandera de gestión de memoria. Después de utilizar el puntero al elemento extraído, éste se tiene que liberar.

Ejemplo:

```
//--- ejemplo de CList::DetachCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   CObject *object=list.DetachCurrent();
   if(object==NULL)
     {
      printf("Error al quitar elemento");
      delete list;
      return;
     }
   //--- utiliza el elemento
   //--- . . .
   //--- borra el elemento
   delete object;
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteCurrent

Borra el elemento de la posición actual de la lista.

```
bool  DeleteCurrent()
```

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede borrar.

Nota

Si la gestión de memoria está activada, la memoria utilizada en los elementos borrados se libera.

Ejemplo:

```
//--- ejemplo de CList::DeleteCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   if(!list.DeleteCurrent())
     {
      printf("Error de eliminación");
      delete list;
      return;
     }
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el elemento de la lista en la posición especificada.

```
bool  Delete(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] Posición donde se desea borrar el elemento de la lista.

Valor devuelto

true si se ejecuta correctamente, false - si el elemento no se puede borrar.

Nota

Si la gestión de memoria está activada, la memoria utilizada en los elementos borrados se libera.

Ejemplo:

```
//--- ejemplo de CList::Delete(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   if(!list.Delete(0))
     {
      printf("Error de eliminación");
      delete list;
      return;
     }
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

Borra todos los elementos de la lista.

```
void  Clear()
```

Nota

Si la gestión de memoria está activada, la memoria utilizada en los elementos borrados se libera.

Ejemplo:

```
//--- ejemplo de CList::Clear()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   //--- borra la lista
   list.Clear();
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

IndexOf

Obtiene el índice del último elemento.

```
int  IndexOf(
   CObject*  element      // Puntero al elemento
   )
```

Parámetros

element

[in] Puntero al elemento de la lista.

Valor devuelto

Índice del ítem de la lista, o -1.

Ejemplo:

```
//--- ejemplo de CList::IndexOf(CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   CObject *object=new CObject;
   if(object==NULL)
     {
      printf("Error en la creación del elemento");
      delete list;
      return;
     }
   if(list.Add(object))
     {
      int pos=list.IndexOf(object);
     }
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetNodeAtIndex

Obtiene el ítem de la lista con el índice especificado.

```
CObject*  GetNodeAtIndex(
   int  pos      // posición
   )
```

Parámetros

pos

[in] Índice del ítem de la lista.

Valor devuelto

Puntero al elemento en caso de éxito, NULL si no se puede obtener el puntero.

Ejemplo:

```
//--- ejemplo de CList::GetNodeAtIndex(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error en la creación del objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   CObject *object=list.GetNodeAtIndex(10);
   if(object==NULL)
     {
      printf("Obtener el error del nodo");
      delete list;
      return;
     }
   //--- utiliza el elemento
   //--- . . .
   //--- no borrar el elemento
   //--- borra la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetFirstNode

Obtiene el primer elemento de la lista.

```
CObject*  GetFirstNode()
```

Valor devuelto

Puntero al primer ítem en caso de éxito, NULL si no se puede obtener el puntero.

Ejemplo:

```
//--- ejemplo de CList::GetFirstNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   CObject *object=list.GetFirstNode();
   if(object==NULL)
     {
      printf("Obtener el error del nodo");
      delete list;
      return;
     }
   //--- utiliza el elemento
   //--- . . .
   //--- no borrar el elemento
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetPrevNode

Obtiene el elemento anterior de la lista.

```
CObject*  GetPrevNode()
```

Valor devuelto

Puntero al elemento anterior, en caso de éxito; NULL si no se puede obtener el puntero.

Ejemplo:

```
//--- ejemplo de CList::GetPrevNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   CObject *object=list.GetPrevNode();
   if(object==NULL)
     {
      printf("Obtener el error del nodo");
      delete list;
      return;
     }
   //--- utiliza el elemento
   //--- . . .
   //--- no borrar el elemento
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetCurrentNode

Obtiene el elemento actual de la lista.

```
CObject*  GetCurrentNode()
```

Valor devuelto

Puntero al elemento actual, si se ejecuta correctamente; NULL si no se puede obtener el puntero.

Ejemplo:

```
//--- ejemplo de CList::GetCurrentNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   CObject *object=list.GetCurrentNode();
   if(object==NULL)
     {
      printf("Obtener el error del nodo");
      delete list;
      return;
     }
   //--- utiliza el elemento
   //--- . . .
   //--- no borrar el elemento
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetNextNode

Obtiene el siguiente ítem de la lista.

```
CObject*  GetNextNode()
```

Valor devuelto

Puntero al siguiente ítem en caso de éxito, NULL si no se puede obtener el puntero.

Ejemplo:

```
//--- ejemplo de CList::GetNextNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   CObject *object=list.GetNextNode();
   if(object==NULL)
     {
      printf("Obtener el error del nodo");
      delete list;
      return;
     }
   //--- utiliza el elemento
   //--- . . .
   //--- no borrar el elemento
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetLastNode

Obtiene el último elemento de la lista.

```
CObject*  GetLastNode()
```

Valor devuelto

Puntero al último elemento en caso de éxito, NULL si no se puede obtener el puntero.

Ejemplo:

```
//--- ejemplo de CList::GetLastNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   CObject *object=list.GetLastNode();
   if(object==NULL)
     {
      printf("Obtener el error del nodo");
      delete list;
      return;
     }
   //--- utiliza el elemento
   //--- . . .
   //--- no borrar el elemento
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Sort

Ordena la lista.

```
void  Sort(
   int  mode      // Modo de ordenación
   )
```

Parámetros

mode

[in] Modo de ordenación.

Valor devuelto

No.

Nota

La lista se ordena siempre en orden ascendente.

Ejemplo:

```
//--- ejemplo de CList::Sort(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- ordenación en modo 0
   list.Sort(0);
   //--- utiliza la lista
   //--- ...
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

MoveToIndex

Mueve el elemento actual de la lista a la posición especificada.

```
bool  MoveToIndex(
   int  pos      // Posición
   )
```

Parámetros

pos

[in] Posición de la lista a mover.

Valor devuelto

true si se ejecuta correctamente, false - si el ítem no se puede mover.

Ejemplo:

```
//--- ejemplo de CList::MoveToIndex(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- mueve el elemento actual al inicio
   list.MoveToIndex(0);
   //--- utiliza la lista
   //--- . . .
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Exchange

Intercambia los elementos de la lista.

```
bool  Exchange(
   CObject*  node1,     // Ítem de la lista
   CObject*  node2      // Ítem de la lista
   )
```

Parámetros

node1

[in] Ítem de la lista

node2

[in] Ítem de la lista

Valor devuelto

true si se ejecuta correctamente, false si los elementos no se pueden intercambiar en algunas posiciones.

Ejemplo:

```
//--- ejemplo de CList::Exchange(CObject*,CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- intercambio
   list.Exchange(list.GetFirstNode(),list.GetLastNode());
   //--- utiliza la lista
   //--- . . .
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareList

Compara la lista con otra.

```
bool  CompareList(
   CList*  list      // Lista con la que llevar a cabo la comparación
   )
```

Parámetros

list

[in] Puntero a una instancia de la clase CList con los elementos fuente para llevar a cabo la comparación.

Valor devuelto

true si las listas son iguales, false - en caso de error.

Ejemplo:

```
//--- ejemplo de CList::CompareList(const CList*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- crea la lista fuente
   CList *src=new CList;
   if(src==NULL)
     {
      printf("Error al crear el objeto");
      delete list;
      return;
     }
   //--- llena la lista fuente
   //--- . . .
   //--- compara con otra lista
   bool result=list.CompareList(src);
   //--- borra las listas
   delete src;
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Búsqueda

Busca un elemento igual a la muestra en la lista ordenada.

```
CObject*  Search(
   CObject*  element      // Muestra
   )
```

Parámetros

element

[in] Elemento a buscar en la lista.

Valor devuelto

Puntero al elemento encontrado si se ejecuta correctamente; NULL, si el ítem no se encuentra.

Ejemplo:

```
//--- ejemplo de CList::Search(CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   //--- ordena la lista
   list.Sort(0);
   //--- crea la muestra
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Error en la creación de la muestra");
      delete list;
      return;
     }
   //--- establece los atributos de la muestra
   //--- . . .
   //--- busca el elemento
   if(list.Search(sample)!=NULL) printf("Elemento encontrado");
   else                          printf("Elemento no encontrado");
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda la lista en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Ejemplo:

```
//--- ejemplo de CList::Save(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- añade elementos a la lista
   //--- . . .
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- elimina la lista
   delete list;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los datos de la lista a partir del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto con la función FileOpen (...)

Valor devuelto

true - si se completa correctamente, false - si hay algún error.

Nota

Cuando se lee el archivo para crear los elementos de la lista se llama al método CList:: CreateElement ().

Ejemplo:

```
//--- ejemplo de CLoad::Load(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- utiliza los elementos de la lista
   //--- . . .
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo de la lista.

```
virtual int  Type()
```

Valor devuelto

Identificador de tipo de la lista (el de CList es 7779).

Ejemplo:

```
//--- ejemplo de CList::Type()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Error al crear el objeto");
      return;
     }
   //--- obtiene el tipo de lista
   int type=list.Type();
   //--- elimina la lista
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

CTreeNode

CTreeNode es una clase de nodos del árbol binario CTree.

Descripción

La clase CTreeNode permite trabajar con nodos del árbol binario CTree. La clase implementa opciones para navegar a través del árbol. Además, también permite trabajar con archivos.

Declaración

```
class CTreeNode : public CObject
```

Título

```
#include <Arrays\TreeNode.mqh>
```

Jerarquía de herencia

CObject

CTreeNode

Descendientes directos

CTree

Métodos de la clase

Atributos	
Owner	Obtiene/establece el puntero del nodo propietario
Left	Obtiene/establece el puntero del nodo izquierdo
Right	Obtiene/establece el puntero del nodo derecho
Balance	Obtiene el peso del nodo
BalanceL	Obtiene el peso de la subrama izquierda del nodo
BalanceR	Obtiene el peso de la subrama derecha del nodo
Creación de un elemento nuevo	
CreateSample	Crea un nuevo nodo
Comparación	
RefreshBalance	Recalcula el peso del nodo
Búsqueda	
GetNext	Obtiene el puntero al siguiente nodo
Entrada/Salida	

© 2000-2025, MetaQuotes Ltd.

Atributos	
SaveNode	Guarda los datos del nodo en el archivo
LoadNode	Descarga los datos del nodo del archivo
virtual Type	Obtiene el identificador de tipo del nodo

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Compare

Los árboles de la clase descendiente CTreeNode tienen aplicaciones prácticas.

Los descendientes de la clase CTreeNode deben tener métodos predefinidos: CreateSample para crear una nueva instancia de la clase descendiente de CTreeNode, Compare para comparar los valores de los campos clave de la clase descendiente de CTreeNode, Type (si hace falta identificar un nodo), SaveNode y LoadNode (si es necesario trabajar con un archivo).

Consideremos ahora un ejemplo de clase descendiente de CTree.

```
//+------------------------------------------------------------------+
//|                                                   MyTreeNode.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                             https://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\TreeNode.mqh>
//+------------------------------------------------------------------+
//| Describe la clase derivada de CTreeNode.                         |
//+------------------------------------------------------------------+
//| Clase CMyTreeNode.                                               |
//| Objetivo: Implementa el nodo de un árbol binario.                |
//|             Descendiente de la clase CTreeNode.                  |
//+------------------------------------------------------------------+
class CMyTreeNode : public CTreeNode
  {
protected:
   //--- datos de usuario
   long              m_long;           // campo clave de tipo long
   double            m_double;         // variable personalizada de tipo double
   string            m_string;         // variable personalizada de tipo string
   datetime          m_datetime;       // variable personalizada de tipo datetime

public:
                     CMyTreeNode();
   //--- métodos para acceder a los datos del usuario
   long              GetLong(void)                { return(m_long); }
   void              SetLong(long value)          { m_long=value;   }
```

© 2000-2025, MetaQuotes Ltd.

```
   double            GetDouble(void)               { return(m_double); }
   void              SetDouble(double value)       { m_double=value;  }
   string            GetString(void)               { return(m_string); }
   void              SetString(string value)       { m_string=value;  }
   datetime          GetDateTime(void)             { return(m_datetime); }
   void              SetDateTime(datetime value)   { m_datetime=value;  }
   //--- métodos para trabajar con archivos
   virtual bool      Save(int file_handle);
   virtual bool      Load(int file_handle);
protected:
   virtual int       Compare(const CObject *node,int mode);
   //--- métodos para crear instancias de clase
   virtual CTreeNode* CreateSample();
  };
//+------------------------------------------------------------------+
//| Constructor de la clase CMyTreeNode.                             |
//| INPUT:  ninguna.                                                 |
//| OUTPUT: ninguna.                                                 |
//| Observaciones: ninguna.                                          |
//+------------------------------------------------------------------+
void CMyTreeNode::CMyTreeNode()
  {
//--- inicialización de los datos de usuario
   m_long        =0;
   m_double      =0.0;
   m_string      ="";
   m_datetime    =0;
  }
//+------------------------------------------------------------------+
//| Comparación con otro nodo por el algoritmo especificado.         |
//| INPUT:  node - elemento a comparar,                              |
//|         mode - identificador del algoritmo de comparación.      |
//| OUTPUT: resultado de la comparación (>0,0,<0).                   |
//| Observaciones: ninguna.                                          |
//+------------------------------------------------------------------+
int CMyTreeNode::Compare(const CObject *node,int mode)
  {
//--- el parámetro mode es ignorado
   int res=0;
//--- conversión de tipos explícita
   CMyTreeNode *n=node;
   res=(int)(m_long-n.m_long);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Creación de una nueva instancia de la clase.                     |
//| INPUT:  ninguna.                                                 |
//| OUTPUT: puntero a una instancia nueva de la clase CMyTreeNode.   |
```

© 2000-2025, MetaQuotes Ltd.

```
//| Observaciones: ninguna.                                          |
//+------------------------------------------------------------------+
CTreeNode* CMyTreeNode::CreateSample()
  {
   CMyTreeNode *result=new CMyTreeNode;
//---
   return(result);
  }
//+------------------------------------------------------------------+
//| Escribe el nodo en el archivo.                                   |
//| INPUT:  file_handle - manejador de archivo abierto para escritura.    |
//| OUTPUT: true si se ejecuta correctamente; false en caso contrario.
//| Observaciones: ninguna.                                          |
//+------------------------------------------------------------------+
bool CMyTreeNode::Save(int file_handle)
  {
   uint i=0,len;
//--- comprobación
   if(file_handle<0) return(false);
//--- escribe los datos de usuario
//--- escribe la variable personalizada de tipo long
   if(FileWriteLong(file_handle,m_long)!=sizeof(long))          return(false);
//--- escribe la variable personalizada de tipo double
   if(FileWriteDouble(file_handle,m_double)!=sizeof(double))    return(false);
//--- escribe la variable personalizada de tipo string
   len=StringLen(m_string);
//--- escribe la longitud de la cadena de caracteres
   if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE)   return(false);
//--- escribe la cadena
   if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- escribe la variable personalizada de tipo datetime
   if(FileWriteLong(file_handle,m_datetime)!=sizeof(long))      return(false);
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Lee los datos del nodo del archivo.                              |
//| INPUT:  file_handle - manejador de archivo abierto para lectura.    |
//| OUTPUT: true si se ejecuta correctamente; false en caso contrario.
//| Observaciones: ninguna.                                          |
//+------------------------------------------------------------------+
bool CMyTreeNode::Load(int file_handle)
  {
   uint i=0,len;
//--- comprobación
   if(file_handle<0) return(false);
//--- lectura
   if(FileIsEnding(file_handle)) return(false);
//--- lee la variable personalizada de tipo char
```

© 2000-2025, MetaQuotes Ltd.

```
//--- lee la variable personalizada de tipo long
   m_long=FileReadLong(file_handle);
//--- lee la variable personalizada de tipo double
   m_double=FileReadDouble(file_handle);
//--- lee la variable personalizada de tipo string
//--- lee la longitud de la cadena
   len=FileReadInteger(file_handle,INT_VALUE);
//--- lee la cadena
   if(len!=0) m_string=FileReadString(file_handle,len);
   else       m_string="";
//--- lee la variable personalizada de tipo datetime
   m_datetime=FileReadLong(file_handle);
//---
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

Owner

Obtiene el puntero del nodo propietario.

```
CTreeNode*  Owner()
```

Valor devuelto

Puntero del nodo propietario.

Owner

Establece el puntero del nodo propietario.

```
void  Owner(
   CTreeNode*  node       // nodo
   )
```

Parámetros

node

[in] Valor nuevo del puntero del nodo propietario.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Left

Obtiene el puntero al nodo izquierdo.

```
CTreeNode*  Left()
```

Valor devuelto

Puntero al nodo izquierdo.

Left

Establece el puntero del nodo izquierdo.

```
void  Left(
   CTreeNode*  node       // nodo
   )
```

Parámetros

node

[in] Valor nuevo del puntero del nodo izquierdo.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Right

Obtiene el puntero del nodo derecho.

```
CTreeNode*  Right()
```

Valor devuelto

El puntero del nodo derecho.

Right

Establece el puntero del nodo derecho.

```
void  Right(
   CTreeNode*  node       // nodo
   )
```

Parámetros

node

[in] Valor nuevo del puntero del nodo derecho.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Balance

Obtiene el peso del nodo.

```
int  Balance() const
```

Valor devuelto

Peso del nodo.

© 2000-2025, MetaQuotes Ltd.

BalanceL

Obtiene el peso de la subrama izquierda del nodo.

```
int  BalanceL() const
```

Valor devuelto

Peso de la subrama izquierda del nodo.

© 2000-2025, MetaQuotes Ltd.

BalanceR

Obtiene el peso de la subrama derecha del nodo.

```
int  BalanceR() const
```

Valor devuelto

Peso de la subrama derecha del nodo.

© 2000-2025, MetaQuotes Ltd.

CreateSample

Crea un nuevo nodo muestra.

```
virtual CTreeNode*  CreateSample()
```

Valor devuelto

Puntero al nodo nuevo o NULL.

© 2000-2025, MetaQuotes Ltd.

RefreshBalance

Recalcula el peso del nodo.

```
int  RefreshBalance()
```

Valor devuelto

Peso del nodo.

© 2000-2025, MetaQuotes Ltd.

GetNext

Obtiene el puntero al siguiente nodo.

```
CTreeNode*  GetNext(
   CTreeNode*  node      // nodo
   )
```

Parámetros

node

[in] Nodo del comienzo de la búsqueda.

Valor devuelto

Puntero al siguiente nodo.

© 2000-2025, MetaQuotes Ltd.

SaveNode

Escribe los datos del nodo en el archivo.

```
bool  SaveNode(
   int  file_handle      // manejador
   )
```

Parámetros

file_handle

[in] Manejador de un archivo binario previamente abierto para escritura.

Valor devuelto

true si se ejecuta correctamente; false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

LoadNode

Lee los datos del nodo a partir del archivo.

```
bool  LoadNode(
   int          file_handle,     // manejador
   CTreeNode*   main             // nodo
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario previamente abierto para lectura.

main

[in] Nodo de datos.

Valor devuelto

true si se ejecuta correctamente; false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo del nodo.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del nodo.

© 2000-2025, MetaQuotes Ltd.

CTree

La clase CTree proporciona un árbol binario de instancias de la clase CTreeNode, e instancias de clases hijas descendientes.

Descripción

La clase CTree permite trabajar con un árbol binario de instancias de CTreeNode, e instancias de clases hijas descendientes. Esta clase implementa opciones para añadir, insertar y borrar elementos del árbol, así como buscarlos. Además de lo anterior, también posibilita trabajar con archivos.

Cabe señalar que CTree no implementa ningún mecanismo para gestionar la memoria dinámica (a diferencia de las clases CList y CArrayObj). Cuando se libera la memoria, todos los nodos del árbol son eliminados.

Declaración

```
class CTree : public CTreeNode
```

Título

```
#include <Arrays\Tree.mqh>
```

Jerarquía de herencia

CObject
  CTreeNode
    CTree

Métodos de la clase

Atributos	
Root	Obtiene el nodo raíz del árbol
Creación de un elemento nuevo	
CreateElement	Crea un nuevo nodo
Relleno	
Insert	Añade un nodo al árbol
Borrado	
Detach	Quita el nodo especificado del árbol
Delete	Borra el nodo especificado del árbol
Clear	Borra todos los elementos del árbol
Búsqueda	
Find	Busca el nodo de acuerdo a la muestra especificada

© 2000-2025, MetaQuotes Ltd.

Atributos	
Entrada/salida	
virtual Save	Guarda los datos del árbol en el archivo
virtual Load	Descarga los datos del árbol del archivo
virtual Type	Obtiene el identificador de tipo del árbol

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CTreeNode

Parent, Parent, Left, Left, Right, Right, Balance, BalanceL, BalanceR, RefreshBalance, GetNext, SaveNode, LoadNode

Árboles de la clase CTreeNode - los descendientes de la clase CTree tienen aplicación práctica.

Los descendientes de la clase CTree deben implementar el método predefinido CreateElement que crea una nueva muestra de la clase descendiente CTreeNode.

Consideremos ahora un ejemplo de clase descendiente de CTree.

```
//+------------------------------------------------------------------+
//|                                                      MyTree.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//| www.metaquotes.net |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\Tree.mqh>
#include "MyTreeNode.mqh"
//---
input int extCountedNodes = 100;
//+------------------------------------------------------------------+
//| Describe la clase CMyTree, descendiente de CTree.                      |
//+------------------------------------------------------------------+
//| Clase CMyTree.                                                   |
//| Objetivo: Construcción de un árbol binario de búsqueda y navegación.    |
//+------------------------------------------------------------------+
class CMyTree : public CTree
  {
public:
   //--- métodos para buscar en el árbol los datos especificados
   CMyTreeNode*         FindByLong(long find_long);
   //--- método que crea un nuevo elemento
   virtual CTreeNode *CreateElement();
  };
//---
```

© 2000-2025, MetaQuotes Ltd.

```
CMyTree MyTree;
//+------------------------------------------------------------------+
//| Creación de un nuevo nodo en el árbol.                           |
//| INPUT:  ninguna.                                                 |
//| Salida: puntero a un nuevo nodo del árbol si se ejecuta correctamente, o NULL.
//| Observaciones: ninguna.                                          |
//+------------------------------------------------------------------+
CTreeNode *CMyTree::CreateElement()
  {
   CMyTreeNode *node=new CMyTreeNode;
//---
   return(node);
  }
//+------------------------------------------------------------------+
//| Búsqueda de un elemento en la lista por valor m_long.            |
//| INPUT:  find_long - valor buscado.                               |
//| OUTPUT: puntero al elemento encontrado en la lista, o NULL.      |
//| Observaciones: ninguna.                                          |
//+------------------------------------------------------------------+
CMyTreeNode* CMyTree::FindByLong(long find_long)
  {
   CMyTreeNode *res=NULL;
   CMyTreeNode *node;
//--- crea un nodo en el árbol para pasar el parámetro de búsqueda
   node=new CMyTreeNode;
   if(node==NULL) return(NULL);
   node.SetLong(find_long);
//---
   res=Find(node);
   delete node;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| script "pruebas con la clase CMyTree"                            |
//+------------------------------------------------------------------+
//---  inicialización de las cadenas de texto
string str_array[11]={"p","oo","iii","uuuu","yyyyy","ttttt","rrrr","eee","ww","q","999
//---
int OnStart() export
  {
   int          i;
   uint         pos;
   int          beg_time,end_time;
   CMyTreeNode *node; //--- puntero temporal a la muestra de la clase CMyTreeNode
//---
   printf("Comienzo de las pruebas %s.",__FILE__);
//--- Llenar MyTree con muestras de la clase MyTreeNode en la cantidad de extCountedNo
   beg_time=GetTickCount();
```

© 2000-2025, MetaQuotes Ltd.

```
   for(i=0;i<extCountedNodes;i++)
     {
      node=MyTree.CreateElement();
      if(node==NULL)
        {
         //--- salida de emergencia
         printf("%s (%4d): error en la creación",__FILE__,__LINE__);
         return(__LINE__);
        }
      NodeSetData(node,i);
      node.SetLong(i);
      MyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("El tiempo de llenado de MyTree es %d ms.",end_time-beg_time);
//--- Crear un árbol temporal TmpMyTree.
   CMyTree TmpMyTree;
//--- Quita el 50% de los elementos del árbol (incluso todos)
//--- añadir elementos al árbol temporal TmpMyTree.
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i+=2)
     {
      node=MyTree.FindByLong(i);
      if(node!=NULL)
         if(MyTree.Detach(node)) TmpMyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("El tiempo de borrado de %d elementos de MyTree es %d ms.",extCountedNodes/2
//--- Devuelve los quitados
   node=TmpMyTree.Root();
   while(node!=NULL)
     {
      if(TmpMyTree.Detach(node)) MyTree.Insert(node);
      node=TmpMyTree.Root();
     }
//--- Comprobar el trabajo del método Save(int file_handle);
   int file_handle;
   file_handle=FileOpen("MiArbol.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!MyTree.Save(file_handle))
        {
         //--- error al escribir en el archivo
         //--- salida de emergencia
         printf("%s: Error %d en %d!",__FILE__,GetLastError(),__LINE__);
         //--- cerrar el archivo antes de terminar!!!
         FileClose(file_handle);
         return(__LINE__);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      FileClose(file_handle);
     }
//--- Comprobar el trabajo del método Load(int file_handle);
   file_handle=FileOpen("MiArbol.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!TmpMyTree.Load(file_handle))
        {
         //--- error al leer del archivo
         //--- salida de emergencia
         printf("%s: Error %d en %d!",__FILE__,GetLastError(),__LINE__);
         //--- cerrar el archivo antes de terminar!!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
     }
//---
   MyTree.Clear();
   TmpMyTree.Clear();
//---
   printf("Finalizar test %s. OK!",__FILE__);
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| Función que genera el contenido del nodo                         |
//+------------------------------------------------------------------+
void NodeToLog(CMyTreeNode *node)
  {
   printf("   %I64d,%f,'%s','%s'",
              node.GetLong(),node.GetDouble(),
              node.GetString(),TimeToString(node.GetDateTime()));
  }
//+------------------------------------------------------------------+
//| Función que llena el nodo con valores aleatorios                 |
//+------------------------------------------------------------------+
void NodeSetData(CMyTreeNode *node,int mode)
  {
   if(mode%2==0)
     {
      node.SetLong(mode*MathRand());
      node.SetDouble(MathPow(2.02,mode)*MathRand());
     }
   else
     {
      node.SetLong(mode*(long)(-1)*MathRand());
      node.SetDouble(-MathPow(2.02,mode)*MathRand());
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   node.SetString(str_array[mode%10]);
   node.SetDateTime(10000*mode);
  }
```

© 2000-2025, MetaQuotes Ltd.

Root

Obtiene el nodo raíz del árbol.

```
CTreeNode*  Root() const
```

Valor devuelto

Puntero al nodo raíz del árbol.

© 2000-2025, MetaQuotes Ltd.

CreateElement

Crea una instancia nueva del nodo.

```
virtual CTreeNode*  CreateElement()
```

Valor devuelto

Puntero a la nueva instancia del nodo, o NULL.

© 2000-2025, MetaQuotes Ltd.

Insert

Añade un nodo al árbol.

```
CTreeNode*  Insert(
   CTreeNode*  new_node       // nodo
   )
```

Parámetros

new_node

[in] puntero al nodo a insertar en el árbol.

Valor devuelto

Puntero al nodo o NULL.

© 2000-2025, MetaQuotes Ltd.

Detach

Quita el nodo especificado del árbol.

```
bool  Detach(
   CTreeNode*  node       // nodo
   )
```

Parámetros

node

[in] Puntero al nodo a quitar.

Valor devuelto

true si se ejecuta correctamente; false en caso contrario.

Nota

Después de quitar el nodo, el puntero no se libera. El árbol está balanceado.

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el nodo especificado del árbol.

```
bool  Delete(
   CTreeNode*  node      // nodo
   )
```

Parámetros

node

[in] Puntero al nodo a borrar.

Valor devuelto

true si se ejecuta correctamente; false en caso contrario.

Nota

Después del borrado se libera el puntero del nodo. El árbol está balanceado.

© 2000-2025, MetaQuotes Ltd.

Clear

Borra todos los nodos del árbol.

```
void  Clear()
```

Valor devuelto

Ninguno.

Nota

Después del borrado se liberan los punteros del nodo.

© 2000-2025, MetaQuotes Ltd.

Find

Busca el nodo en el árbol de acuerdo a la muestra.

```
CTreeNode*  Find(
   CTreeNode*  node      // nodo
   )
```

Parámetros

node

[in] Nodo que contiene los datos de muestra para realizar la búsqueda.

Valor devuelto

Puntero al nodo encontrado o NULL.

© 2000-2025, MetaQuotes Ltd.

Save

Escribe los datos del árbol en el archivo.

```
virtual bool  Save(
   int  file_handle      // manejador
   )
```

Parámetros

file_handle

[in] Manejador de un archivo binario previamente abierto para escritura.

Valor devuelto

true si se ejecuta correctamente; false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Load

Lee los datos del árbol a partir del archivo.

```
virtual bool  Load(
   int  file_handle      // manejador
   )
```

Parámetros

file_handle

[in] Manejador a un archivo binario previamente abierto para lectura.

Valor devuelto

true si se ejecuta correctamente; false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo del árbol.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del árbol.

© 2000-2025, MetaQuotes Ltd.

Colecciones de datos genéricas

La biblioteca contiene clases e interfaces para determinar las colecciones genéricas que, a su vez, dan al ususario la posibilidad de crear coleciones rigurosamente tipadas. Estas hacen el trabajo con los datos más cómodo y productivo que en las colecciones tipadas habituales.

La biblioteca se ubica en el catálogo de trabajo del terminal en la carpeta Include\Generic.

Objetos:

Objeto	Descripción	Tipo
ICollection	Interfaz para la implementación de colecciones de datos genéricas	INTERFACE
IEqualityComparable	Interfaz para implementar objetos que se puedan comparar entre sí	INTERFACE
IComparable	Interfaz para implementar objetos que se puedan comparar entre sí mediante la relación "más, menos o igual"	INTERFACE
IComparer	Interfaz para implementar una clase universal que compara dos objetos del tipo T sobre la relación "más, menos o igual"	INTERFACE
IEqualityComparer	Interfaz para implementar una clase universal que compara dos objetos del tipo T sobre la igualdad	INTERFACE
IList	Interfaz para la implementantación de listas de datos genéricas	INTERFACE
IMap	Interfaz para la implementantación de colecciones genéricas de parejas "clave - valor"	INTERFACE
ISet	Interfaz para la implementación de conjuntos de datos genéricos	INTERFACE
CDefaultComparer	Clase auxiliar que implementa la interfaz genérica IComparer<T> sobre la base de los métodos globales Compare	CLASS
CDefaultEqualityComparer	Clase auxiliar que implementa la interfaz genérica IEqualityComparer<T> con los métodos globales Equals<T> y GetHashCode	CLASS
CArrayList	Clase genérica que implementa la interfaz IList<T>	CLASS

© 2000-2025, MetaQuotes Ltd.

Objeto	Descripción	Tipo
CKeyValuePair	La clase implementa la pareja "clave - valor"	CLASS
CHashMap	Clase genérica que implementa la interfaz IMap<TKey, TValue>	CLASS
CHashSet	Clase genérica que implementa la interfaz ISet<T>	CLASS
CLinkedListNode	Clase auxiliar necesaria para implementar la clase CLinkedListNode<T>	CLASS
CLinkedList	Clase genérica que implementa la interfaz ICollection<T>	CLASS
CQueue	Clase genérica que implementa la interfaz ICollection<T>	CLASS
CRedBlackTreeNode	Clase auxiliar necesaria para implementar la clase CRedBlackTree<T>	CLASS
CRedBlackTree	Clase genérica que implementa la interfaz ICollection<T>	CLASS
CSortedMap	Clase genérica que implementa la interfaz IMap<TKey, TValue>	CLASS
CSortedSet	Clase genérica que implementa la interfaz ISet<T>	CLASS
CStack	Clase genérica que implementa la interfaz ICollection<T>	CLASS

Metodos globales:

Método	Descripción
ArrayBinarySearch	Busca el valor indicado en una matriz unidimensional clasificada en orden ascendente, usando la interfaz IComparable<T> para comparar los elementos.
ArrayIndexOf	Busca la primera aparición de un valor en una matriz unidimensional
ArrayLastIndexOf	Busca la última aparición de un valor en una matriz unidimensional
ArrayReverse	Cambia la secuenciación de los elementos en una matriz unidimensional
Compare	Compara dos valores sobre la relación "Mayor, menor o igual"

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
Equals	Compara dos valores sobre su igualdad
GetHashCode	Calcula el valor del código hash

© 2000-2025, MetaQuotes Ltd.

ICollection<T>

ICollection<T> es una interfaz para la implementación de colecciones de datos genéricas.

Descripción

La interfaz ICollection<T> determina los métodos principales para el trabajo con colecciones: el cálculo del número de elementos, la limpieza de la colección, la adición y eliminación de elementos y otros.

Declaración

```
template<typename T>
interface ICollection
```

Encabezamiento

```
#include <Generic\Interfaces\ICollection.mqh>
```

Jerarquía de herencia

ICollection

Descendientes directos

CLinkedList, CQueue, CRedBlackTree, CStack, IList, IMap, ISet

Métodos de clase

Método	Descripción
Add	Añade un elemento a la colección
Count	Retorna el número de elementos en la colección
Contains	Determina si la colección contiene el elemento con el valor indicado
CopyTo	Copia todos los elementos de la colección a la matriz especificada, comenzando por un índice en particular
Clear	Elimina todos los elementos de la colección
Remove	Elimina la primera aparición del elemento especificado de la colección

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento a la colección.

```
bool Add(
   T  value     // valor del elemento
   );
```

Parámetros

value

[in] Valor del elemento a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Count

Retorna el número de elementos en la colección.

```
int Count();
```

Valor devuelto

Retorna el número de elementos.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si la colección contiene el elemento con el valor indicado.

```
bool Contains(
   T  item     // valor buscado
   );
```

Parámetros

item

[in] Valor buscado.

Valor devuelto

Retorna true, si en la colección existe el elemento con el valor indicado, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todos los elementos de la colección a la matriz especificada, comenzando por un índice en particular.

```
int CopyTo(
   T&          dst_array[],     // matriz para el guardado
   const int   dst_start=0      // índice inicial para el guardado
   );
```

Parámetros

&dst_array[]

[out] Matriz en la que se guardarán los elementos de la colección.

dst_start=0

[in] Índice en la matriz desde el que se comienza el copiado.

Valor devuelto

Retorna el número de elementos copiados.

© 2000-2025, MetaQuotes Ltd.

Clear

Elimina todos los elementos de la colección.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Elimina la primera aparición del elemento especificado de la colección.

```
bool Remove(
   T  item     // valor del elemento
   );
```

Parámetros

item

[in] Valor del elemento que se debe eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

IEqualityComparable<T>

La interfaz IEqualityComparable<T> es una interfaz para implementar objetos que se pueden comparar entre sí.

Descripción

La interfaz IEqualityComparable<T> determina los métodos para obtener el código hash del objeto actual y compararlo con otro objeto del mismo tipo sobre la igualdad.

Declaración

```
template<typename T>
interface IEqualityComparable
```

Encabezamiento

```
#include <Generic\Interfaces\IEqualityComparable.mqh>
```

Jerarquía de herencia

IEqualityComparable

Descendientes directos

IComparable

Métodos de clase

Método	Descripción
Equals	Compara el objeto actual con el valor especificado
HashCode	Calcula el valor del código hash para el objeto actual.

© 2000-2025, MetaQuotes Ltd.

Equals

Compara el objeto actual con el valor especificado.

```
bool Equals(
   T  value     // valor para la comparación
   );
```

Parámetros

value

[in] Valor con el que se compara el objeto actual.

Valor devuelto

Retorna true si los objetos son iguales, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

HashCode

Calcula el valor del código hash para el objeto actual.

```
int HashCode();
```

Valor devuelto

Retorna el código hash.

© 2000-2025, MetaQuotes Ltd.

IComparable<T>

La interfaz IComparable<T> es una interfaz para implementar objetos que se pueden comparar entre sí mediante la relación "más, menos o igual".

Descripción

La interfaz IComparable<T> determina el método de comparación del objeto actual con otro del mismo tipo, sobre cuya base se puede clasificar la colección de estos objetos.

Declaración

```
template<typename T>
interface IComparable : public IEqualityComparable<T>
```

Encabezamiento

```
#include <Generic\Interfaces\IComparable.mqh>
```

Jerarquía de herencia

IEqualityComparable

IComparable

Descendientes directos

CKeyValuePair

Métodos de clase

Método	Descripción
Compare	Compara el objeto actual con el valor especificado

© 2000-2025, MetaQuotes Ltd.

Compare

Compara el objeto actual con el valor especificado.

```
int Compare(
   T  value     // valor para la comparación
   );
```

Parámetros

value

[in] Valor con el que se compara el objeto actual.

Valor devuelto

Retorna un número que expresa la relación del objeto actual y el transmitido:

- resultado menor a cero — el objeto actual es menor al transmitido
- resultado igual a cero — el objeto actual es igual al transmitido
- resultado superior a cero — el objeto actual es superior al transmitido

© 2000-2025, MetaQuotes Ltd.

IComparer<T>

La interfaz IComparable<T> es una interfaz para implementar una clase universal que compara dos objetos del tipo T entre sí mediante la relación "más, menos o igual".

Descripción

La interfaz IComparer<T> determina el método de comparación de dos objetos del tipo T, sobre cuya base se puede clasificar la colección de estos objetos.

Declaración

```
template<typename T>
interface IComparer
```

Encabezamiento

```
#include <Generic\Interfaces\IComparer.mqh>
```

Jerarquía de herencia

IComparer

Descendientes directos

CDefaultComparer

Métodos de clase

Método	Descripción
Compare	Compara dos valores del tipo T

© 2000-2025, MetaQuotes Ltd.

Compare

Compara dos valores del tipo T.

```
int Compare(
   T  x,      // primer valor
   T  y       // segundo valor
   );
```

Parámetros

x

[in] Primer valor para la comparación.

y

[in] Primer valor para la comparación.

Valor devuelto

Retorna un número que expresa la relación de los dos valores comparados:

- resultado menor a cero — x menor a y (x<y)
- resultado igual a cero — x igual a y (x=y)
- resultado mayor a cero — x mayor a y (x>y)

© 2000-2025, MetaQuotes Ltd.

IEqualityComparer<T>

La interfaz IEqualityComparer<T> es una interfaz para implementar una clase universal que compara dos objetos del tipo T.

Descripción

La interfaz IEqualityComparer<T> determina los métodos para obtener el código hash de un objeto del tipo T y comparar dos objetos del tipo T sobre la igualdad.

Declaración

```
template<typename T>
interface IEqualityComparer
```

Encabezamiento

```
#include <Generic\Interfaces\IEqualityComparer.mqh>
```

Jerarquía de herencia

IEqualityComparer

Descendientes directos

CDefaultEqualityComparer

Métodos de clase

Método	Descripción
Equals	Compara dos valores del tipo T
HashCode	Calcula el valor del código hash a partir de un objeto del tipo T

© 2000-2025, MetaQuotes Ltd.

Equals

Compara dos valores del tipo T.

```
bool Equals(
   T  x,      // primer valor
   T  y       // segundo valor
   );
```

Parámetros

x

[in] Primer valor para la comparación.

y

[in] Segundo valor para la comparación.

Valor devuelto

Retorna true si los valores son iguales, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

HashCode

Calcula el valor del código hash a partir de un objeto del tipo T.

```
int HashCode(
   T  value     // objeto para el cálculo
   );
```

Parámetros

value

[in] Objeto para el que se debe obtener el código hash.

Valor devuelto

Retorna el código hash.

© 2000-2025, MetaQuotes Ltd.

IList<T>

La interfaz IList<T> es una interfaz para la implementación de listas de datos genéricas.

Descripción

La interfaz IList<T> determina los métodos principales para trabajar con una lista: acceso al elemento según el índice, búsqueda y eliminación del elemento, clasificación y otros.

Declaración

```
template<typename T>
interface IList : public ICollection<T>
```

Encabezamiento

```
#include <Generic\Interfaces\IList.mqh>
```

Jerarquía de herencia

ICollection

IList

Descendientes directos

CArrayList

Métodos de clase

Método	Descripción
TryGetValue	Obtiene un elemento de la lista según el índice establecido
TrySetValue	Cambia un valor de la lista según el índice establecido
Insert	Inserta un elemento en la lista según el índice especificado
IndexOf	Busca la primera aparición de un valor en la lista
LastIndexOf	Busca la última aparición de un valor en la lista
RemoveAt	Elimina un elemento de la lista según el índice especificado

© 2000-2025, MetaQuotes Ltd.

TryGetValue

Obtiene un elemento de la lista según el índice establecido.

```
bool TryGetValue(
   const int  index,      // índice del elemento
   T&         value       // variable para el guardado
   );
```

Parámetros

index

[in] Índice del elemento de la lista.

&value

[out] Variable en la que se guardará el valor indicado del elemento de la lista.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

Cambia un valor de la lista según el índice establecido.

```
bool TrySetValue(
   const int  index,      // índice del elemento
   T          value       // valor nuevo
   );
```

Parámetros

index

[in] Índice del elemento de la lista.

value

[in] Nuevo valor que se deber asignar al elemento indicado de la lista.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta un elemento en la lista según el índice especificado.

```
bool Insert(
   const int  index,      // índice a insertar
   T          item        // valor a insertar
   );
```

Parámetros

index

[in] Índice a insertar.

item

[in] Valor que se debe insertar según el índice indicado.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

IndexOf

Busca la primera aparición de un valor en la lista.

```
int IndexOf(
   T  item     // valor buscado
   );
```

Parámetros

item

[in] Valor buscado.

Valor devuelto

Retorna el índice del primer elemento encontrado. Si no se localiza el valor, retorna -1.

© 2000-2025, MetaQuotes Ltd.

LastIndexOf

Busca la última aparición de un valor en la lista.

```
int LastIndexOf(
   T  item     // valor buscado
   );
```

Parámetros

item

[in] Valor buscado.

Valor devuelto

Retorna el índice del último elemento encontrado. Si no se localiza el valor, retorna -1.

© 2000-2025, MetaQuotes Ltd.

RemoveAt

Elimina un elemento de la lista según el índice especificado.

```
bool RemoveAt(
   const int  index     // índice del elemento
   );
```

Parámetros

index

[in] Índice del elemento que se debe eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

IMap<TKey, TValue>

La interfaz IMap<TKey, TValue> es una interfaz para la implementación de colecciones genéricas de las parejas "clave — valor".

Descripción

La interfaz IMap<TKey, TValue> determina los métodos principales para trabajar con colecciones cuyos datos se guardan en forma de pareja "clave - valor".

Declaración

```
template<typename TKey, typename TValue>
interface IMap : public ICollection<TKey>
```

Encabezamiento

```
#include <Generic\Interfaces\IMap.mqh>
```

Jerarquía de herencia

ICollection

IMap

Descendientes directos

CHashMap, CSortedMap

Métodos de clase

Método	Descripción
Add	Añade una pareja "clave — valor" a la colección
Contains	Determina si la colección contiene la pareja "clave — valor" con la clave y valor indicados
Remove	Elimina la primera aparición de la pareja "clave — valor" con la clave indicada de la colección
TryGetValue	Obtiene un elemento de la colección según la clave indicada
TrySetValue	Cambia el valor de una pareja "clave — valor" de una colección según la clave indicada
CopyTo	Copia todas las parejas "clave — valor" de una colección a las matrices indicadas, comenzando por un índice determinado

© 2000-2025, MetaQuotes Ltd.

Add

Añade una pareja "clave — valor" a una colección.

```
bool Add(
   TKey    key,       // clave
   TValue  value      // valor
   );
```

Parámetros

key

[in] Clave.

value

[in] Valor.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si la colección contiene la pareja "clave — valor" con la clave y valor indicados.

```
bool Contains(
   TKey    key,       // clave
   TValue  value      // valor
   );
```

Parámetros

key

[in] Clave.

value

[in] Valor.

Valor devuelto

Retorna true, si en la colección existe la pareja "clave - valor" con la clave y el valor establecidos, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

Remove

Elimina la primera aparición de la pareja "clave — valor" con la clave indicada de la colección.

```
bool Remove(
   TKey  key     // clave
   );
```

Parámetros

key

[in] Clave.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

TryGetValue

Obtiene un elemento de la colección según la clave indicada.

```
bool TryGetValue(
   TKey     key,       // clave
   TValue&  value      // variable para guardar el valor
   );
```

Parámetros

key

[in] Clave.

&value

[out] Variable en la que se guardará el valor indicado de la pareja "clave — valor".

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

Cambia el valor de una pareja "clave — valor" de una colección según la clave indicada.

```
bool TrySetValue(
   TKey    key,       // clave
   TValue  value      // nuevo valor
   );
```

Parámetros

key

[in] Clave.

value

[in] Nuevo valor que se deber asignar a la pareja "clave — valor" indicada

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todas las parejas "clave — valor" de una colección a las matrices indicadas, comenzando por un índice determinado.

```
int CopyTo(
   TKey&       dst_keys[],       // matriz para guardar las claves
   TValue&     dst_values[],     // matriz para guardar los valores
   const int   dst_start=0       // índice inicial para el guardado
   );
```

Parámetros

&dst_keys[]

[out] Matriz en la que se guardarán todas las claves de la colección.

&dst_values[]

[out] Matriz en la que se guardarán todos los valores de las claves correspondientes de la colección.

dst_start=0

[in] Índice en las matrices desde el que se comienza el copiado.

Valor devuelto

Retorna el número de parejas "clave - valor" copiadas.

© 2000-2025, MetaQuotes Ltd.

ISet<T>

La interfaz ISet<T> es una interfaz para la implementación de conjuntos genéricos de datos.

Descripción

La interfaz ISet<T> determina los métodos principales para trabajar con conjuntos: unión e intersección de conjuntos, definición de subjuntos rigurosos y no rigurosos y otros.

Declaración

```
template<typename T>
interface ISet : public ICollection<T>
```

Encabezamiento

```
#include <Generic\Interfaces\ISet.mqh>
```

Jerarquía de herencia

ICollection

ISet

Descendientes directos

CHashSet, CSortedSet

Métodos de clase

Método	Descripción
ExceptWith	Realiza la operación de diferencia de la colección (matriz) actual y la transmitida
IntersectWith	Realiza la operación de intersección de la colección (matriz) actual y la transmitida
SymmetricExceptWith	Realiza la operación de diferencia simétrica de la colección (matriz) actual y la transmitida
UnionWith	Realiza la operación de unión de la colección (matriz) actual y la transmitida
IsProperSubsetOf	Determina si el conjunto actual es un subconjunto estricto de la colección o matriz establecidas
IsProperSupersetOf	Determina si el conjunto actual es un superconjunto estricto de la colección o matriz establecidas
IsSubsetOf	Determina si el conjunto actual es un subconjunto de la colección o matriz establecidas
IsSupersetOf	Determina si el conjunto actual es un superconjunto de la colección o matriz establecidas

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
Overlaps	Determina si el conjunto actual se cruza con la colección o matriz establecidas
SetEquals	Determina si el conjunto actual contiene todos los elementos de la colección o matriz establecidas

© 2000-2025, MetaQuotes Ltd.

ExceptWith

Realiza la operación de diferencia de la colección (matriz) actual y la transmitida. Es decir, elimina de la colección (matriz) actual todos los elementos que también existen en la colección (matriz) indicada.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void ExceptWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección que se sustraerá del conjunto actual.

&collection[]

[in] Matriz que se sustraerá del conjunto actual.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

IntersectWith

Realiza la operación de intersección de la colección (matriz) actual y la transmitida. Es decir, deja en la colección (matriz) actual solo los elementos que también existen en la colección (matriz) indicada.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void IntersectWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección con la que se construirá la multiplicación.

&collection[]

[in] Matriz con la que se construirá la multiplicación.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

Realiza la operación de diferencia simétrica de la colección (matriz) actual y la transmitida. Es decir, en la colección (matriz) actual solo se contendrán aquellos elementos que existían o bien en el objeto fuente, o bien en el transmitido, pero no simultáneamente en ambos.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void SymmetricExceptWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección con la que se construirá la diferencia simétrica.

&collection[]

[in] Matriz con la que se construirá la diferencia simétrica.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

UnionWith

Realiza la operación de unión de la colección (matriz) actual y la transmitida. Es decir, añade a la colección (matriz) actual los elementos ausentes desde la colección (matriz) indicada.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void UnionWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección con la que se construirá la suma.

&collection[]

[in] Matriz con la que se construirá la suma.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

Determina si el conjunto actual es un subconjunto estricto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsProperSubsetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto actual es un subconjunto estricto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsProperSupersetOf

Determina si el conjunto actual es un superconjunto estricto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsProperSupersetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto actual es un superconjunto estricto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsSubsetOf

Determina si el conjunto actual es un subconjunto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsSubsetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto actual es un subconjunto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

Determina si el conjunto actual es un superconjunto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsSupersetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto actual es un superconjunto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

Overlaps

Determina si el conjunto actual se cruza con la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // colección para la comparación
   );
```

Versión para trabajar con la matriz.

```
bool Overlaps(
   T&  array[]                     // matriz para la comparación
   );
```

Parámetros

**collection*

[in] Colección para definir la intersección.

&collection[]

[in] Matriz para definir la intersección.

Valor devuelto

Retorna true si entre el conjunto actual y la colección o matriz existe una intersección, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SetEquals

Determina si el conjunto actual contiene todos los elementos de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // colección para la comparación
   );
```

Versión para trabajar con la matriz.

```
bool SetEquals(
   T&  array[]                     // matriz para la comparación
   );
```

Parámetros

**collection*

[in] Colección para la comparación de elementos.

&collection[]

[in] Colección para la comparación de elementos.

Valor devuelto

Retorna true si en el conjunto actual existen todos los elementos de la colección o matriz indicada, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CDefaultComparer<T>

La clase CDefaultComparer<T> es una clase auxiliar que implementa la interfaz genérica IComparer<T> basándose en los métodos globales Compare.

Descripción

La clase CDefaultComparer<T> se usa por defecto en las colecciones de datos genéricas, si el usuario no utiliza explícitamente otra clase que implemente la interfaz IComparer<T>.

Declaración

```
template<typename T>
class CDefaultComparer : public IComparer<T>
```

Encabezamiento

```
#include <Generic\Internal\DefaultComparer.mqh>
```

Jerarquía de herencia

IComparer

CDefaultComparer

Métodos de clase

Método	Descripción
Compare	Compara dos valores del tipo T

© 2000-2025, MetaQuotes Ltd.

Compare

Compara dos valores del tipo T.

```
int Compare(
   T  x,      // primer valor
   T  y       // segundo valor
   );
```

Parámetros

x

[in] Primer valor para la comparación.

y

[in] Segundo valor para la comparación.

Valor devuelto

Retorna un número que expresa la relación de los dos valores comparados:

- resultado menor a cero — x menor a y (x<y)
- resultado igual a cero — x igual a y (x=y)
- resultado mayor a cero — x mayor a y (x>y)

Nota

La comparación de los valores x e y tiene lugar basándose una de las sobrecargas del método global Compare, dependiendo del tipo T.

© 2000-2025, MetaQuotes Ltd.

CDefaultEqualityComparer<T>

La clase CDefaultEqualityComparer<T> es una clase auxiliar que implementa la interfaz genérica IEqualityComparer<T> basándose en los métodos globales de Equals<T> y GetHashCode.

Descripción

La clase CDefaultEqualityComparer<T> se usa por defecto en las colecciones de datos genéricas, si el usuario no utiliza explícitamente otra clase que implemente la interfaz IEqualityComparer<T>.

Declaración

```
template<typename T>
class CDefaultEqualityComparer : public IEqualityComparer<T>
```

Encabezamiento

```
#include <Generic\Internal\DefaultEqualityComparer.mqh>
```

Jerarquía de herencia

IEqualityComparer

CDefaultEqualityComparer

Métodos de clase

Método	Descripción
Equals	Compara dos valores del tipo T
HashCode	Calcula el valor del código hash a partir de un objeto del tipo T

© 2000-2025, MetaQuotes Ltd.

Equals

Compara dos valores del tipo T.

```
bool Equals(
   T  x,      // primer valor
   T  y       // segundo valor
   );
```

Parámetros

x

[in] Primer valor para la comparación.

y

[in] Segundo valor para la comparación.

Valor devuelto

Retorna true si los valores son iguales, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

HashCode

Calcula el valor del código hash a partir de un objeto del tipo T.

```
int HashCode(
   T  value     // objeto para el cálculo
   );
```

Parámetros

value

[in] Objeto para el que se debe obtener el código hash.

Valor devuelto

Retorna el código hash.

© 2000-2025, MetaQuotes Ltd.

CRedBlackTreeNode<T>

La clase CRedBlackTreeNode<T> es una clase auxiliar necesaria para implementar la clase CRedBlackTree<T>.

Descripción

La clase CRedBlackTreeNode<T> es un nodo del árbol rojo-negro CRedBlackTree<T>. En la clase se han implementado métodos para navegar por el árbol.

Declaración

```
template<typename T>
class CRedBlackTreeNode
```

Encabezamiento

```
#include <Generic\RedBlackTree.mqh>
```

Métodos de clase

Método	Descripción
Value	Retorna y establece el valor del nodo
Parent	Retorna y establece el puntero al nodo padre
Left	Retorna y establece el puntero al nodo izquierdo
Right	Retorna y establece el puntero al nodo derecho
Color	Retorna y establece el color del nodo
IsLeaf	Determina si el nodo especificado es una hoja
CreateEmptyNode	Crea un nuevo nodo de color negro sin ancestros ni descendientes y retorna el puntero al mismo

© 2000-2025, MetaQuotes Ltd.

Value (método Get)

Retorna el valor del nodo.

```
T Value();
```

Valor devuelto

Retorna el valor del nodo.

Value (método Set)

Establece el valor del nodo.

```
void Value(
   T  value     // valor del nodo
   );
```

Parámetros

value

[in] Valor del nodo.

© 2000-2025, MetaQuotes Ltd.

Parent (método Get)

Retorna el puntero al nodo padre

```
CRedBlackTreeNode<T>* Parent();
```

Valor devuelto

Retorna el puntero al nodo padre

Parent (método Set)

Establece el puntero al nodo padre

```
void Parent(
   CRedBlackTreeNode<T>*  node     // puntero al nodo padre
   );
```

Parámetros

node

[in] Puntero al nodo padre.

© 2000-2025, MetaQuotes Ltd.

Left (método Get)

Retorna el puntero al nodo izquierdo

```
CRedBlackTreeNode<T>* Left();
```

Valor devuelto

Retorna el puntero al nodo izquierdo

Left (método Set)

Establece el puntero al nodo izquierdo

```
void Left(
   CRedBlackTreeNode<T>*  node     // puntero al nodo izquierdo
   );
```

Parámetros

node

[in] Puntero al nodo izquierdo.

© 2000-2025, MetaQuotes Ltd.

Right (método Get)

Retorna el puntero al nodo derecho

```
CRedBlackTreeNode<T>* Right();
```

Valor devuelto

Retorna el puntero al nodo derecho

Right (método Set)

Establece el puntero al nodo derecho.

```
void Right(
   CRedBlackTreeNode<T>*  node     // puntero al nodo derecho
   );
```

Parámetros

node

[in] Puntero al nodo derecho.

© 2000-2025, MetaQuotes Ltd.

Color (método Get)

Retorna el color del nodo.

```
ENUM_RED_BLACK_TREE_NODE_TYPE Color();
```

Valor devuelto

Retorna el color del nodo.

Color (método Set)

Establece el color del nodo

```
void Color(
   ENUM_RED_BLACK_TREE_NODE_TYPE  clr     // color del nodo
   );
```

Parámetros

clr

[in] Color del nodo.

Nota

El color del nodo se establece con un valor de la enumeración ENUM_RED_BLACK_TREE_NODE_TYPE y tiene dos tipos:

- RED_BLACK_TREE_NODE_RED — color rojo del nodo;
- RED_BLACK_TREE_NODE_BLACK — color negro del nodo.

© 2000-2025, MetaQuotes Ltd.

IsLeaf

Determina si el nodo especificado es una hoja.

```
bool IsLeaf();
```

Valor devuelto

Retorna true, si el nodo es una hoja, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CreateEmptyNode

Crea un nuevo nodo de color negro sin ancestros ni descendientes y retorna el puntero al mismo.

```
static CRedBlackTreeNode<T>* CreateEmptyNode();
```

Valor devuelto

Retorna el puntero al nuevo nodo.

© 2000-2025, MetaQuotes Ltd.

CLinkedListNode<T>

La clase CLinkedListNode<T> es una clase auxiliar imprescindible para implementar la clase CLinkedListNode<T>.

Descripción

La clase CLinkedListNode<T> es un nodo de la lista bidireccional CLinkedListNode<T>. En la clase se han implementado métodos para navegar por la lista.

Declaración

```
template<typename T>
class CLinkedListNode
```

Encabezamiento

```
#include <Generic\LinkedList.mqh>
```

Métodos de clase

Método	Descripción
List	Retorna y establece el puntero a la lista bidireccional CLinkedList<T>
Next	Retorna y establece el puntero al siguiente nodo
Previous	Retorna y establece el puntero al nodo anterior
Value	Retorna y establece el valor del nodo

© 2000-2025, MetaQuotes Ltd.

List (método Get)

Retorna el puntero a la lista bidireccional CLinkedList<T>

```
CLinkedList<T>* List();
```

Valor devuelto

Retorna el puntero a la lista bidireccional CLinkedList<T>.

List (método Set)

Establece el puntero a la lista bidireccional CLinkedList<T>.

```
void List(
   CLinkedList<T>*  value     // puntero a la lista
   );
```

Parámetros

value

[in] Puntero a la lista bidireccional CLinkedList<T>.

© 2000-2025, MetaQuotes Ltd.

Next (método Get)

Retorna el puntero al siguiente nodo.

```
CLinkedListNode<T>* Next();
```

Valor devuelto

Retorna el puntero al siguiente nodo.

Next (método Set)

Establece el puntero al siguiente nodo.

```
void Next(
   CLinkedListNode<T>*  value     // puntero al siguiente nodo
   );
```

Parámetros

value

[in] Puntero al siguiente nodo.

© 2000-2025, MetaQuotes Ltd.

Previous (método Get)

Retorna el puntero al nodo anterior.

```
CLinkedListNode<T>* Previous();
```

Valor devuelto

Retorna el puntero al nodo anterior.

Previous (método Set)

Establece el puntero al nodo anterior.

```
void Previous(
   CLinkedListNode<T>*  value     // puntero al nodo anterior
   );
```

Parámetros

value

[in] Puntero al nodo anterior.

© 2000-2025, MetaQuotes Ltd.

Value (método Get)

Retorna el valor del nodo.

```
T Value();
```

Valor devuelto

Retorna el valor del nodo.

Value (método Set)

Establece el valor del nodo.

```
void Value(
   T  value     // valor del nodo
   );
```

Parámetros

value

[in] Valor del nodo.

© 2000-2025, MetaQuotes Ltd.

CKeyValuePair<TKey,TValue>

La clase CKeyValuePair<TKey,TValue> es una clase que implementa la pareja "clave — valor".

Descripción

La clase CKeyValuePair<TKey,TValue> implementa los métodos de trabajo directo con la clave y el valor de la pareja "clave — valor".

Declaración

```
template<typename TKey, typename TValue>
class CKeyValuePair : public IComparable<CKeyValuePair<TKey,TValue>*>
```

Encabezamiento

```
#include <Generic\HashMap.mqh>
```

Jerarquía de herencia

IEqualityComparable

IComparable

CKeyValuePair

Métodos de clase

Método	Descripción
Key	Retorna y establece la clave de la pareja "clave — valor"
Value	Retorna y establece el valor de la pareja "clave — valor"
Clone	Crea una nueva pareja "clave — valor", cuya clave y valor son iguales a los actuales
Compare	Compara la pareja "clave — valor" actual con la indicada
Equals	Compara la pareja "clave — valor" actual con la indicada para comprobar la igualdad
HashCode	Calcula el valor del código hash a partir la pareja "clave — valor"

© 2000-2025, MetaQuotes Ltd.

Key (método Get)

Retorna la clave de la pareja "clave — valor"

```
TKey Key();
```

Valor devuelto

Retorna la clave.

Key (método Set)

Establece el valor de la pareja "clave — valor"

```
void Key(
   TKey  key     // clave
   );
```

Parámetros

key

[in] Clave.

© 2000-2025, MetaQuotes Ltd.

Value (método Get)

Retorna el valor de la pareja "clave — valor"

```
TValue Value();
```

Valor devuelto

Retorna el valor.

Value (método Set)

Establece el valor de la pareja "clave — valor"

```
void Value(
   TValue  value     // valor
   );
```

Parámetros

value

[in] Valor

© 2000-2025, MetaQuotes Ltd.

Clone

Crea una nueva pareja "clave — valor", cuya clave y valor son iguales a los actuales.

```
TValue>* Clone();
```

Valor devuelto

Retorna una nueva pareja "clave - valor"

© 2000-2025, MetaQuotes Ltd.

Compare

Compara la pareja "clave — valor" actual con la indicada.

```
int Compare(
   CKeyValuePair<TKeyTValue>*  pair     // pareja para la comparación
   );
```

Parámetros

pair

[in] Pareja para la comparación.

Valor devuelto

Retorna una cifra que expresa la relación de la pareja actual "clave — valor" y la transmitida:

- resultado menor a cero — la pareja "clave — valor" actual es menor a la transmitida
- resultado igual a cero — la pareja "clave — valor" actual es igual a la transmitida
- resultado mayor a cero — la pareja "clave — valor" actual es mayor a la transmitida

Nota

La comparación de las dos parejas "clave — valor" tiene lugar según la clave.

© 2000-2025, MetaQuotes Ltd.

Equals

Compara la pareja "clave — valor" actual con el objeto de igualdad establecido.

```
bool Equals(
   CKeyValuePair<TKeyTValue>*  pair     // pareja para la comparación
   );
```

Parámetros

pair

[in] Pareja para la comparación

Valor devuelto

Retorna true, si las parejas "clave-valor" son iguales, de lo contrario, false.

Nota

La comparación de las dos parejas "clave — valor" tiene lugar según la clave.

© 2000-2025, MetaQuotes Ltd.

HashCode

Calcula el valor del código hash de la pareja "clave — valor".

```
int HashCode();
```

Valor devuelto

Retorna el código hash.

Nota

El código hash de la pareja "clave — valor" es igual al código hash de la clave.

© 2000-2025, MetaQuotes Ltd.

CArrayList<T>

La clase CArrayList<T> es una clase genérica que implementa la interfaz IList<T>.

Descripción

La clase CArrayList<T> es la implementación de una lista dinámica de datos de tipo T. Esta clase proporciona los métodos principales para trabajar con una lista: acceso al elemento según el índice, búsqueda y eliminación del elemento, clasificación y otros.

Declaración

```
template<typename T>
class CArrayList : public IList<T>
```

Encabezamiento

```
#include <Generic\ArrayList.mqh>
```

Jerarquía de herencia

ICollection
  IList
    CArrayList

Métodos de clase

Método	Descripción
Capacity	Retrona y establece la capacidad actual de la lista
Count	Retorna el número de elementos en la lista
Contains	Determina si la lista contiene el elemento con el valor indicado.
TrimExcess	Reduce la capacidad de la lista hasta un número real de elementos
TryGetValue	Recibe un elemento de la lista según el índice indicado
TrySetValue	Establece el valor de un elemento de la lista según el índice indicado
Add	Añade un elemento a la lista
AddRange	Añade una colección o una matriz de elementos a la lista
Insert	Inserta un elemento en la lista según el índice especificado
InsertRange	Inserta una colección o un conjunto de elementos en la lista según el índice especificado
CopyTo	Copia todos los elementos de la lista a la matriz especificada, comenzando por un índice en particular
BinarySearch	Busca un valor especificado en una lista ordenada de manera ascendente

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
IndexOf	Busca la primera aparición de un valor en la lista
LastIndexOf	Busca la última aparición de un valor en la lista
Clear	Elimina todos los elementos de la colección
Remove	Elimina la primera aparición de un elemento especificado de la lista
RemoveAt	Elimina un elemento especificado de la lista según el índice dado
RemoveRange	Elimina un rango de elementos de la lista
Reverse	Cambia la secuenciación de los elementos en la lista
Sort	Clasifica los elementos de la lista

© 2000-2025, MetaQuotes Ltd.

Capacity (método Get)

Retorna la capacidad actual de la lista.

```
int Capacity();
```

Valor devuelto

Retorna la capacidad actual de la lista.

Capacity (método Set)

Retrona y establece la capacidad total de la lista

```
void Capacity(
   const int  capacity     // valor de la capacidad
   );
```

Parámetros

capacity

[in] Nuevo valor de la capacidad.

© 2000-2025, MetaQuotes Ltd.

Count

Retorna el número de elementos en la lista.

```
int Count();
```

Valor devuelto

Retorna el número de elementos.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si la lista contiene el elemento con el valor indicado.

```
bool Contains(
   T  item     // valor buscado
   );
```

Parámetros

item

[in] Valor buscado.

Valor devuelto

Retorna true, si en la lista existe el elemento con el valor indicado, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

Reduce la capacidad de la lista hasta un número real de elementos, liberando de esta forma la memoria no usada.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

TryGetValue

Recibe un elemento de la lista según el índice indicado.

```
bool TryGetValue(
   const int  index,      // índice
   T&         value       // variable para el guardado
   );
```

Parámetros

index

[in] Índice del elemento de la lista cuyo valor se debe obtener.

&value

[out] Variable para guardar el valor del elemento.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

Establece el valor de un elemento de la lista según el índice indicado.

```
bool TrySetValue(
   const int  index,      // índice
   T          value       // valor del elemento
   );
```

Parámetros

index

[in] Índice del elemento de la lista cuyo valor se debe establecer.

value

[in] Valor a esteblecer del elemento de la lista.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento a la lista.

```
bool Add(
   T  value     // valor del elemento
   );
```

Parámetros

value

[in] Valor del elemento a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

AddRange

Añade una colección o matriz de elementos a la lista.

Versión para añadir matrices.

```
bool AddRange(
   const T&  array[]               // matriz para la adición
   );
```

Versión para añadir colecciones.

```
bool AddRange(
   ICollection<T>*  collection     // colección a añadir
   );
```

Parámetros

&array[]

[in] Matriz a añadir.

**collection*

[in] Colección a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta un elemento en la lista según el índice especificado.

```
bool Insert(
   const int  index,      // índice a insertar
   T          item        // valor a insertar
   );
```

Parámetros

index

[in] Índice a insertar.

item

[in] Valor que se debe insertar según el índice indicado.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

InsertRange

Inserta una colección o un conjunto de elementos en la lista según el índice especificado.

Versión para insertar una matriz.

```
bool InsertRange(
   const int  index,               // índice a insertar
   const T&   array[]              // matriz a insertar
   );
```

Versión para la inserción de la colección.

```
bool InsertRange(
   const int        index,         // índice a insertar
   ICollection<T>*  collection     // colección a insertar
   );
```

Parámetros

index

[in] Índice a insertar.

&array[]

[in] Matriz que se debe insertar según el índice indicado.

**collection*

[in] Colección que se debe insertar según el índice indicado.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todos los elementos de la lista a la matriz especificada, comenzando por un índice en particular.

```
int CopyTo(
   T&          dst_array[],     // matriz para el guardado
   const int   dst_start=0      // índice inicial para el guardado
   );
```

Parámetros

&dst_array[]

[out] Matriz en la que se guardarán los elementos de búsqueda.

dst_start=0

[in] Índice en la matriz desde el que se comienza el copiado.

Valor devuelto

Retorna el número de elementos copiados.

© 2000-2025, MetaQuotes Ltd.

BinarySearch

Busca un valor especificado en una lista ordenada de manera ascendente.

Versión para la búsqueda con un rango de valores indicado, usando, además, una clase que implementa la interfaz IComparable<T> para comparar elementos.

```
int BinarySearch(
   const int       index,        // índice inicial
   const int       count,        // rango de búsqueda
   T               item,         // valor buscado
   IComparer<T>*   comparer      // interfaz para la comparación
   );
```

Versión para la búsqueda, con uso de una clase que implementa la interfaz IComparable<T> para comparar elementos.

```
int BinarySearch(
   T               item,         // valor buscado
   IComparer<T>*   comparer      // interfaz para la comparación
   );
```

Versión para la búsqueda, con uso del método global ::Compare para comparar elementos.

```
int BinarySearch(
   T  item                       // valor buscado
   );
```

Parámetros

index

[in] Índice inicial desde el que comienza la búsqueda.

count

[in] Longitud del rango de búsqueda.

item

[in] Valor buscado.

**comparer*

[in] Interfaz para comparar los elementos.

Valor devuelto

Retorna el índice del elemento encontrado. Si el valor buscado no se localiza, retorna el índice del menor elemento que se encuentre más cerca según su valor.

© 2000-2025, MetaQuotes Ltd.

IndexOf

Busca la primera aparición de un valor en la lista.

Versión para buscar por toda la lista.

```
int IndexOf(
   T  item                    // valor buscado
   );
```

Versión para buscar desde la posición indicada y hasta el final de la lista.

```
int IndexOf(
   T           item,          // valor buscado
   const int   start_index    // índice inicial
   );
```

Versión para buscar desde la posición indicada en el rango indicador.

```
int IndexOf(
   T           item,          // valor buscado
   const int   start_index,   // índice inicial
   const int   count          // rango de búsqueda
   );
```

Parámetros

item

[in] Valor buscado.

start_index

[in] Índice inicial desde el que comienza la búsqueda.

count

[in] Longitud del rango de búsqueda.

Valor devuelto

Retorna el índice del primer elemento encontrado. Si no se localiza el valor, retorna -1.

© 2000-2025, MetaQuotes Ltd.

LastIndexOf

Busca la última aparición de un valor en la lista.

Versión para buscar por toda la lista.

```
int LastIndexOf(
   T  item                    // valor buscado
   );
```

Versión para buscar desde la posición indicada y hasta el final de la lista.

```
int LastIndexOf(
   T          item,           // valor buscado
   const int  start_index     // índice inicial
   );
```

Versión para buscar desde la posición indicada en el rango indicador.

```
int LastIndexOf(
   T          item,           // valor buscado
   const int  start_index,    // índice inicial
   const int  count           // rango de búsqueda
   );
```

Parámetros

item

[in] Valor buscado.

start_index

[in] Índice inicial desde el que comienza la búsqueda.

count

[in] Longitud del rango de búsqueda.

Valor devuelto

Retorna el índice del último elemento encontrado. Si no se localiza el valor, retorna -1.

© 2000-2025, MetaQuotes Ltd.

Clear

Elimina todos los elementos de la colección.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Elimina la primera aparición de un elemento especificado de la lista.

```
bool Remove(
   T  item     // valor del elemento
   );
```

Parámetros

item

[in] Valor del elemento que se debe eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

RemoveAt

Elimina un elemento especificado de la lista según el índice dado.

```
bool RemoveAt(
   const int  index     // índice
   );
```

Parámetros

index

[in] Índice del elemento a eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

RemoveRange

Elimina un rango de elementos de la lista.

```
bool RemoveRange(
   const int  start_index,     // índice inicial
   const int  count            // número de elementos
   );
```

Parámetros

start_index

[in] Índice inicial desde el que se comienza la eliminación.

count

[in] Número de elementos a eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Reverse

Cambia la secuenciación de los elementos en la lista.

Versión para trabajar con toda la lista.

```
bool Reverse();
```

Versión para trabajar con un rango establecido de elementos de la lista.

```
bool Reverse(
   const int  start_index,     // índice inicial
   const int  count            // número de elementos
   );
```

Parámetros

start_index

[in] Índice inicial.

count

[in] Número de elementos de la lista que participan en la operación.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Sort

Clasifica los elementos de la lista.

Versión para clasificar todos los elementos de la lista.

```
bool Sort();
```

Versión para clasificar todos los elementos de la lista, con uso de una clase que implementa la interfaz IComparable<T> para comparar elementos.

```
bool Sort(
   IComparer<T>*  comparer         // interfaz para la comparación
   );
```

Versión para clasificar un rango de elementos establecido de la lista, con uso de una clase que implementa la interfaz IComparable<T> para comparar elementos.

```
bool Sort(
   const int      start_index,     // índice inicial
   const int      count,           // número de elementos
   IComparer<T>*  comparer         // interfaz para la comparación
   );
```

Parámetros

**comparer*

[in] Interfaz para comparar los elementos.

start_index

[in] Índice inicial desde el que se comienza la clasificación.

count

[in] Longitud del rango de clasificación.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

CHashMap<TKey,TValue>

La clase CHashMap<TKey, TValue> es una clase genérica que implementa la interfaz IMap<TKey, TValue>.

Descripción

La clase CHashMap<TKey, TValue> es la implementación de un recuadro hash dinámico cuyos datos se guardan en forma de parejas desordenadas "clave — valor", respetando la necesidad de unicidad de la clave. Esta clase proporciona los métodos esenciales para trabajar con el recuadro hash: acceso al valor según la clave, búsqueda y eliminación de una pareja "clave-valor" y otros.

Declaración

```
template<typename TKey, typename TValue>
class CHashMap : public IMap<TKey,TValue>
```

Encabezamiento

```
#include <Generic\HashMap.mqh>
```

Jerarquía de herencia

ICollection

IMap

CHashMap

Métodos de clase

Método	Descripción
Add	Añade una pareja "clave — valor" a un recuadro hash
Count	Retorna el número de elementos en el recuadro hash
Comparer	Retorna el puntero a la interfaz IEqualityComparer<T>, usada para organizar el recuadro hash
Contains	Determina si el recuadro hash contiene la pareja "clave — valor" indicada
ContainsKey	Determina si el recuadro hash contiene la pareja "clave — valor" con la clave indicada
ContainsValue	La clase CHashMap<TKey, TValue> es una clase genérica que implementa la interfaz IMap<TKey, TValue>
CopyTo	Copia todas las parejas "clave — valor" de un recuadro hash a las matrices indicadas, comenzando por un índice determinado
Clear	Elimina todos los elementos del recuadro hash
Remove	Encuentra la primera aparición de una pareja "clave — valor" del recuadro hash

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
TryGetValue	Obtiene un elemento del recuadro hash según la clave indicada
TrySetValue	Cambia el valor de una pareja "clave — valor" del recuadro hash según la clave indicada

© 2000-2025, MetaQuotes Ltd.

Add

Añade una pareja "clave — valor" al recuadro hash.

Versión para añadir la pareja "clave — valor" formada.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // pareja "clave — valor"
   );
```

Versión para añadir una nueva pareja "clave — valor" con la clave y valor indicados.

```
bool Add(
   TKey    key,                         // clave
   TValue  value                        // valor
   );
```

Parámetros

**pair*

[in] Pareja "clave — valor".

key

[in] Clave.

value

[in] Valor.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Count

Retorna el número de elementos en el recuadro hash.

```
int Count();
```

© 2000-2025, MetaQuotes Ltd.

Comparer

Retorna el puntero a la interfaz IEqualityComparer<T>, usada para organizar el recuadro hash.

```
IEqualityComparer<TKey>* Comparer() const;
```

Valor devuelto

Retorna el puntero a la interfaz IEqualityComparer<T>.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si el recuadro hash contiene la pareja "clave — valor" indicada.

Versión para trabajar con la pareja "clave — valor" formada.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // pareja "clave — valor"
   );
```

Versión para trabajar con una pareja "clave — valor" en forma de clave y valor establecidos por separado.

```
bool Contains(
   TKey   key,                          // clave
   TValue value                         // valor
   );
```

Parámetros

**item*

[in] Pareja "clave — valor".

key

[in] Clave.

value

[in] Valor.

Valor devuelto

Retorna true, si en el recuadro hash existe una pareja "clave-valor" con la clave y el valor establecidos, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ContainsKey

Determina si el recuadro hash contiene la pareja "clave — valor" con la clave indicada.

```
bool ContainsKey(
   TKey  key     // clave
   );
```

Parámetros

key

[in] Clave.

Valor devuelto

Retorna true, si en el recuadro hash existe la pareja "clave-valor" con la clave establecida, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ContainsValue

Determina si el recuadro hash contiene la pareja "clave — valor" con el valor indicado.

```
bool ContainsValue(
   TValue  value     // valor
   );
```

Parámetros

value

[in] Valor.

Valor devuelto

Retorna true, si en el recuadro hash existe la pareja "clave-valor" con el valor establecido, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todas las parejas "clave — valor" de un recuadro hash a las matrices indicadas, comenzando por un índice determinado.

Versión para copiar un recuadro hash a una matriz de parejas "clave — valor".

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // matriz para guardar las parejas "c
   const int                    dst_start=0      // índice inicial para el guardado
   );
```

Versión para copiar un recuadro hash a matrices aparte para claves y valores.

```
int CopyTo(
   TKey&       dst_keys[],                       // matriz para grabar las claves
   TValue&     dst_values[],                     // matriz para grabar los valores
   const int   dst_start=0                       // índice inicial para el guardado
   );
```

Parámetros

&dst_array[]

[out] Matriz en la que se guardarán todas las parejas del recuadro hash.

&dst_keys[]

[out] Matriz en la que se guardarán todas las claves del recuadro hash.

&dst_values[]

[out] Matriz en la que se guardarán todos los valores del recuadro hash.

dst_start=0

[in] Índice de la matriz desde el que se comienza el copiado.

Valor devuelto

Retorna el número de parejas "clave - valor" copiadas.

© 2000-2025, MetaQuotes Ltd.

Clear

Elimina todos los elementos del recuadro hash.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Encuentra la primera aparición de una pareja "clave — valor" de un recuadro hash.

Versión para eliminar una pareja "clave — valor" según la pareja "clave — valor" formada.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // pareja "clave — valor"
   );
```

Versión para eliminar una pareja "clave — valor" según la clave.

```
bool Remove(
   TKey  key                            // clave
   );
```

Parámetros

**item*

[in] Pareja "clave — valor".

key

[in] Clave.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

TryGetValue

Obtiene un elemento del recuadro hash según la clave indicada.

```
bool TryGetValue(
   TKey      key,       // clave
   TValue&   value      // variable para guardar el valor
   );
```

Parámetros

key

[in] Clave.

&value

[out] Variable en la que se guardará el valor indicado de la pareja "clave — valor".

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

Cambia el valor de una pareja "clave — valor" de un recuadro hash según la clave indicada.

```
bool TrySetValue(
   TKey    key,       // clave
   TValue  value      // nuevo valor
   );
```

Parámetros

key

[in] Clave.

value

[in] Nuevo valor que se deber asignar a la pareja "clave — valor" indicada.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

CHashSet<T>

La clase CHashSet<T> es una clase genérica que implementa la interfaz ISet<T>.

Descripción

La clase CHashSet<T> es la implementación de un conjunto dinámico no ordenado de datos del tipo T, respetando las exigencias de unicidad de cada valor. Esta clase proporciona los métodos principales para trabajar con conjuntos, y las operaciones con ellos: unificación y enumeración de conjuntos, definición de subconjuntos rigurosos y no rigurosos y otros.

Declaración

```
template<typename T>
class CHashSet : public ISet<T>
```

Encabezamiento

```
#include <Generic\HashSet.mqh>
```

Jerarquía de herencia

ICollection
  ISet
    CHashSet

Métodos de clase

Método	Descripción
Add	Añade un elemento al conjunto
Count	Retorna el número de elementos en el conjunto
Comparer	Determina si el conjunto contiene el elemento con el valor indicado
Contains	Retorna el puntero a la interfaz IEqualityComparer<T>, usada para organizar el conjunto
TrimExcess	Reduce la capacidad de un conjunto hasta un número real de elementos, liberando de esta forma la memoria no usada.
CopyTo	Copia todos los elementos de un conjunto a la matriz especificada, comenzando por un índice en particular
Clear	Elimina todos los elementos de un conjunto
Remove	Elimina un elemento indicado de un conjunto
ExceptWith	Realiza la operación de diferencia de la colección (matriz) actual y la transmitida
IntersectWith	Realiza la operación de intersección de la colección (matriz) actual y la transmitida

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
SymmetricExceptWith	Realiza la operación de diferencia simétrica de la colección (matriz) actual y la transmitida
UnionWith	Realiza la operación de unión de la colección (matriz) actual y la transmitida
IsProperSubsetOf	Determina si el conjunto actual es un subconjunto estricto de la colección o matriz establecidas
IsProperSupersetOf	Determina si el conjunto actual es un superconjunto estricto de la colección o matriz establecidas
IsSubsetOf	Determina si el conjunto actual es un subconjunto de la colección o matriz establecidas
IsSupersetOf	Determina si el conjunto actual es un superconjunto de la colección o matriz establecidas
Overlaps	Determina si el conjunto actual se cruza con la colección o matriz establecidas
SetEquals	Determina si el conjunto actual contiene todos los elementos de la colección o matriz establecidas

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al conjunto.

```
bool Add(
   T  value     // valor del elemento
   );
```

Parámetros

value

[in] Valor del elemento a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Count

Retorna el número de elementos en el conjunto.

```
int Count();
```

Valor devuelto

Retorna el número de elementos.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si el conjunto contiene el elemento con el valor indicado.

```
bool Contains(
   T  item     // valor buscado
   );
```

Parámetros

item

[in] Valor buscado.

Valor devuelto

Retorna true, si en el conjunto existe un elemento con el valor indicado, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

Comparer

Retorna el puntero a la interfaz IEqualityComparer<T>, usada para organizar el conjunto.

```
IEqualityComparer<T>* Comparer() const;
```

Valor devuelto

Retorna el puntero a la interfaz IEqualityComparer<T>.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

Reduce la capacidad de un conjunto hasta un número real de elementos, liberando de esta forma la memoria no usada.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todos los elementos de un conjunto a la matriz especificada, comenzando por un índice en particular.

```
int CopyTo(
   T&          dst_array[],     // matriz para el guardado
   const int   dst_start=0      // índice inicial para el guardado
   );
```

Parámetros

&dst_array[]

[out] Matriz en la que se guardarán los elementos del conjunto.

dst_start=0

[in] Índice en la matriz desde el que se comienza el copiado.

Valor devuelto

Retorna el número de elementos copiados.

© 2000-2025, MetaQuotes Ltd.

Clear

Elimina todos los elementos de un conjunto.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Elimina un elemento indicado de un conjunto.

```
bool Remove(
   T  item     // valor del elemento
   );
```

Parámetros

item

[in] Valor del elemento que se debe eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

ExceptWith

Realiza la operación de diferencia de la colección (matriz) actual y la transmitida. Es decir, elimina de la colección (matriz) actual todos los elementos que también existen en la colección (matriz) indicada.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void ExceptWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección que se sustraerá del conjunto actual.

&collection[]

[in] Matriz que se sustraerá del conjunto actual.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

IntersectWith

Realiza la operación de intersección de la colección (matriz) actual y la transmitida. Es decir, deja en la colección (matriz) actual solo los elementos que también existen en la colección (matriz) indicada.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void IntersectWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección con la que se construirá la multiplicación.

&collection[]

[in] Matriz con la que se construirá la multiplicación.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

Realiza la operación de diferencia simétrica de la colección (matriz) actual y la transmitida. Es decir, en la colección (matriz) actual solo se contendrán aquellos elementos que existían o bien en el objeto fuente, o bien en el transmitido, pero no simultáneamente en ambos.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void SymmetricExceptWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección con la que se construirá la diferencia simétrica.

&collection[]

[in] Matriz con la que se construirá la diferencia simétrica.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

UnionWith

Realiza la operación de unión de la colección (matriz) actual y la transmitida. Es decir, añade a la colección (matriz) actual los elementos ausentes desde la colección (matriz) indicada.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void UnionWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección con la que se construirá la suma.

&collection[]

[in] Matriz con la que se construirá la suma.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

Determina si el conjunto actual es un subconjunto estricto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsProperSubsetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto actual es un subconjunto estricto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsProperSupersetOf

Determina si el conjunto actual es un superconjunto estricto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsProperSupersetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto actual es un superconjunto estricto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsSubsetOf

Determina si el conjunto actual es un subconjunto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsSubsetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto actual es un subconjunto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

Determina si el conjunto actual es un superconjunto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsSupersetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto actual es un superconjunto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

Overlaps

Determina si el conjunto actual se cruza con la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // colección para la comparación
   );
```

Versión para trabajar con la matriz.

```
bool Overlaps(
   T&  array[]                     // matriz para la comparación
   );
```

Parámetros

**collection*

[in] Colección para definir la intersección.

&collection[]

[in] Matriz para definir la intersección.

Valor devuelto

Retorna true si entre el conjunto actual y la colección o matriz existe una intersección, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SetEquals

Determina si el conjunto actual contiene todos los elementos de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // colección para la comparación
   );
```

Versión para trabajar con la matriz.

```
bool SetEquals(
   T&  array[]                     // matriz para la comparación
   );
```

Parámetros

**collection*

[in] Colección para la comparación de elementos.

&collection[]

[in] Matriz para la comparación de elementos.

Valor devuelto

Retorna true si en el conjunto actual existen todos los elementos de la colección o matriz indicada, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CLinkedList<T>

La clase CLinkedList<T> es una clase genérica que implementa la interfaz ICollection<T>.

Descripción

La clase CLinkedList<T> es la implementación de lista dinámica bidireccional de datos del tipo T. Esta clase proporciona los métodos principales para trabajar con listas bidireccionales: adición, eliminación, búsqueda del elemento y otros.

Declaración

```
template<typename T>
class CLinkedList : public ICollection<T>
```

Encabezamiento

```
#include <Generic\LinkedList.mqh>
```

Jerarquía de herencia

ICollection

CLinkedList

Métodos de clase

Método	Descripción
Add	Añade un elemento a la lista bidireccional
AddAfter	Añade un elemento a la lista bidireccional después del nodo establecido
AddBefore	Añade un elemento a la lista bidireccional antes del nodo establecido
AddFirst	Añade un elemento al inicio de la lista bidireccional
AddLast	Añade un elemento al final de la lista bidireccional
Count	Retorna el número de elementos en la lista bidireccional
Head	Retorna el puntero al primer nodo de la lista bidireccional
First	Retorna el puntero al primer nodo de la lista bidireccional
Last	Retorna el puntero al último nodo de la lista bidireccional
Contains	Determina si la lista bidireccional contiene el elemento con el valor indicado
CopyTo	Copia todos los elementos de la lista bidireccional a la matriz especificada, comenzando por un índice en particular
Clear	Elimina todos los elementos de la colección
Remove	Elimina la primera aparición del elemento especificado de la lista bidireccional

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
RemoveFirst	Elimina el primer elemento de la lista bidireccional
RemoveLast	Elimina el último elemento de la lista bidireccional
Find	Busca la primera aparición del valor establecido en la lista bidireccional.
FindLast	Busca la última aparición del valor establecido en la lista bidireccional.

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento a la lista bidireccional.

```
bool Add(
   T  value     // valor del elemento
   );
```

Parámetros

value

[in] Valor del elemento a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

AddAfter

Añade un elemento a la lista bidireccional después del nodo establecido.

Versión para añadir un elemento según el valor.

```
CLinkedListNode<T>* AddAfter(
   CLinkedListNode<T>*  node,         // nodo tras el cual se da la adición
   T                    value         // elemento a añadir
   );
```

Valor devuelto

Retorna el puntero al nodo añadido.

Versión para añadir un elemento como nodo formado según un valor.

```
bool AddAfter(
   CLinkedListNode<T>*  node,         // nodo tras el cual se da la adición
   CLinkedListNode<T>*  new_node      // nodo añadido
   );
```

Parámetros

node

[in] Nodo de la lista bidireccional tras el cual se añadirá un nuevo elemento.

value

[in] Elemento a añadir.

**new_node*

[in] Nodo a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

AddBefore

Añade un elemento a la lista bidireccional antes del nodo establecido.

Versión para añadir un elemento según el valor.

```
CLinkedListNode<T>* AddBefore(
   CLinkedListNode<T>*  node,         // nodo antes del cual se da la adición
   T                    value         // elemento a añadir
   );
```

Valor devuelto

Retorna el puntero al nodo añadido.

Versión para añadir un elemento como nodo formado según un valor.

```
bool AddBefore(
   CLinkedListNode<T>*  node,         // nodo antes del cual se da la adición
   CLinkedListNode<T>*  new_node      // nodo añadido
   );
```

Parámetros

node

[in] Nodo de la lista bidireccional antes del cual se añadirá un nuevo elemento.

value

[in] Elemento a añadir.

**new_node*

[in] Nodo a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

AddFirst

Añade un elemento al inicio de la lista bidireccional.

Versión para añadir un elemento según el valor.

```
CLinkedListNode<T>* AddFirst(
   T    value                     // elemento a añadir
   );
```

Valor devuelto

Retorna el puntero al nodo añadido.

Versión para añadir un elemento como nodo formado según un valor.

```
bool AddFirst(
   CLinkedListNode<T>*  node      // nodo a añadir
   );
```

Parámetros

value

[in] Elemento a añadir.

**node*

[in] Nodo a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

AddLast

Añade un elemento al final de la lista bidireccional.

Versión para añadir un elemento según el valor.

```
CLinkedListNode<T>* AddLast(
   T  value                        // elemento a añadir
   );
```

Valor devuelto

Retorna el puntero al nodo añadido.

Версия для добавления элемента, как сформированного узла по значению.

```
bool AddLast(
   CLinkedListNode<T>*  node      // nodo a añadir
   );
```

Parámetros

value

[in] Elemento a añadir.

**node*

[in] Nodo a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Count

Retorna el número de elementos en la lista bidireccional.

```
int Count();
```

Valor devuelto

Retorna el número de elementos.

© 2000-2025, MetaQuotes Ltd.

Head

Retorna el puntero al primer nodo de la lista bidireccional.

```
CLinkedListNode<T>* Head();
```

Valor devuelto

Retorna el puntero al primer nodo.

© 2000-2025, MetaQuotes Ltd.

First

Retorna el puntero al primer nodo de la lista bidireccional.

```
CLinkedListNode<T>* First();
```

Valor devuelto

Retorna el puntero al primer nodo.

© 2000-2025, MetaQuotes Ltd.

Last

Retorna el puntero al último nodo de la lista bidireccional.

```
CLinkedListNode<T>* Last();
```

Valor devuelto

Retorna el puntero al último nodo.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si la lista bidireccional contiene el elemento con el valor indicado.

```
bool Contains(
   T  item     // valor buscado
   );
```

Parámetros

item

[in] Valor buscado.

Valor devuelto

Retorna true, si en la lista bidireccional existe un elemento con el valor indicado, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todos los elementos de la lista bidireccional a la matriz especificada, comenzando por un índice en particular.

```
int CopyTo(
   T&          dst_array[],     // matriz para el guardado
   const int   dst_start=0      // índice inicial para el guardado
   );
```

Parámetros

&dst_array[]

[out] Matriz en la que se guardarán los elementos de la lista bidireccional.

dst_start=0

[in] Índice en la matriz desde el que se comienza el copiado.

Valor devuelto

Retorna el número de elementos copiados.

© 2000-2025, MetaQuotes Ltd.

Clear

Elimina todos los elementos de la colección.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Elimina la primera aparición del elemento especificado de la lista bidireccional.

Versión para eliminar un elemento según el valor.

```
bool Remove(
   T  item                        // valor del elemento
   );
```

Versión para eliminar un elemento según el puntero a un nodo.

```
bool Remove(
   CLinkedListNode<T>*  node      // nodo del elemento
   );
```

Parámetros

item

[in] Valor del elemento que se debe eliminar.

**node*

[in] Nodo del elemento que se debe eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

RemoveFirst

Elimina el primer elemento de la lista bidireccional.

```
bool RemoveFirst();
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

RemoveLast

Elimina el último elemento de la lista bidireccional.

```
bool RemoveLast();
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Find

Busca la primera entrada del valor establecido en la lista bidireccional.

```
CLinkedListNode<T>* Find(
   T  value     // valor buscado
   );
```

Parámetros

value

[in] Valor buscado.

Valor devuelto

Retorna el puntero al primer nodo localizado que contenga el valor buscado en el caso de éxito, de lo contrario, NULL.

© 2000-2025, MetaQuotes Ltd.

FindLast

Busca la última aparición del valor establecido en la lista bidireccional.

```
CLinkedListNode<T>* FindLast(
   T  value     // valor buscado
   );
```

Parámetros

value

[in] Valor buscado.

Valor devuelto

Retorna el puntero al último nodo localizado que contenga el valor buscado en el caso de éxito, de lo contrario, NULL.

© 2000-2025, MetaQuotes Ltd.

CQueue<T>

La clase CQueue<T> es una clase genérica que implementa la interfaz ICollection<T>.

Descripción

La clase CQueue<T> es una colección dinámica de datos del tipo T, organizada en forma de series que funciona según el principio "primera entrada — primera salida" (FIFO).

Declaración

```
template<typename T>
class CQueue : public ICollection<T>
```

Encabezamiento

```
#include <Generic\Queue.mqh>
```

Jerarquía de herencia

ICollection

CQueue

Métodos de clase

Método	Descripción
Add	Añade un elemento a la fila
Enqueue	Añade un elemento a la fila
Count	Retorna el número de elementos en la fila
Contains	Determina si la fila contiene el elemento con el valor indicado
TrimExcess	Reduce la capacidad de la fila hasta un número real de elementos, liberando de esta forma la memoria no usada.
CopyTo	Copia todos los elementos de la fila a la matriz especificada, comenzando por un índice en particular
Clear	Elimina todos los elementos de una fila
Remove	Elimina la primera aparición del elemento especificado de la fila
Dequeue	Retorna el elemento inicial, eliminándolo de la fila
Peek	Retorna el elemento inicial, sin eliminarlo de la fila

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento a la fila.

```
bool Add(
   T  value     // valor del elemento
   );
```

Parámetros

value

[in] Valor del elemento a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Enqueue

Añade un elemento a la fila.

```
bool Enqueue(
   T  value     // elemento a añadir
   );
```

Parámetros

value

[in] Elemento a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Count

Retorna el número de elementos en la fila.

```
int Count();
```

Valor devuelto

Retorna el número de elementos.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si la fila contiene el elemento con el valor indicado.

```
bool Contains(
   T  item     // valor buscado
   );
```

Parámetros

item

[in] Valor buscado.

Valor devuelto

Retorna true, si en la fila existe un elemento con el valor indicado, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

Reduce la capacidad de la fila hasta un número real de elementos, liberando de esta forma la memoria no usada.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todos los elementos de la fila a la matriz especificada, comenzando por un índice en particular.

```
int CopyTo(
   T&          dst_array[],     // matriz para el guardado
   const int   dst_start=0      // índice inicial para el guardado
   );
```

Parámetros

&dst_array[]

[out] Matriz en la que se guardarán los elementos de la fila.

dst_start=0

[in] Índice en la matriz desde el que se comienza el copiado.

Valor devuelto

Retorna el número de elementos copiados.

© 2000-2025, MetaQuotes Ltd.

Clear

Elimina todos los elementos de una fila.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Elimina la primera aparición del elemento especificado de la fila.

```
bool Remove(
   T  item     // valor del elemento
   );
```

Parámetros

item

[in] Valor del elemento que se debe eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Dequeue

Retorna el elemento inicial, eliminándolo de la fila.

```
T Dequeue();
```

Valor devuelto

Retorna el elemento inicial.

© 2000-2025, MetaQuotes Ltd.

Peek

Retorna el elemento inicial, sin eliminarlo de la fila.

```
T Peek();
```

Valor devuelto

Retorna el elemento inicial.

© 2000-2025, MetaQuotes Ltd.

CRedBlackTree<T>

La clase CRedBlackTree<T> es una clase genérica que implementa la interfaz ICollection<T>.

Descripción

La clase CRedBlackTree<T> es la implementación de un árbol rojo-negro dinámico en cuyos nodos se guardan datos del tipo T. La clase proporciona los métodos principales para trabajar con árboles rojo-negros: adición, eliminación, búsqueda del valor máximo, mínimo y otros.

Declaración

```
template<typename T>
class CRedBlackTree : public ICollection<T>
```

Encabezamiento

```
#include <Generic\RedBlackTree.mqh>
```

Jerarquía de herencia

ICollection

CRedBlackTree

Métodos de clase

Método	Descripción
Add	Añade un elemento al árbol rojo-negro
Root	Retorna el puntero a la raíz del árbol rojo-negro
Count	Retorna el número de elementos en el árbol rojo-negro
Contains	Determina si el árbol rojo-negro contiene el elemento con el valor indicado
Comparer	Retorna el puntero a la interfaz IComparer<T>, usada para organizar el árbol rojo-negro
TryGetMin	Obtiene el elemento mínimo del árbol rojo-negro
TryGetMax	Obtiene el elemento máximo del árbol rojo-negro
CopyTo	Copia todos los elementos del árbol rojo-negro a la matriz especificada, comenzando por un índice en particular
Clear	Elimina todos los elementos del árbol rojo-negro
Remove	Elimina la aparición del elemento indicado del árbol rojo-negro
RemoveMin	Elimina todos los elementos con el valor mínimo del árbol rojo-negro
RemoveMax	Elimina todos los elementos con el valor máximo del árbol rojo-negro
Find	Busca la aparición del elemento establecido en el árbol rojo-negro

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
FindMax	Busca el elemento con el valor máximo del árbol rojo-negro
FindMin	Busca el elemento con el valor mínimo del árbol rojo-negro

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al árbol rojo-negro.

```
bool Add(
   T  value     // elemento a añadir
   );
```

Parámetros

value

[in] Elemento a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Count

Retorna el número de elementos en el árbol rojo-negro.

```
int Count();
```

Valor devuelto

Retorna el número de elementos.

© 2000-2025, MetaQuotes Ltd.

Root

Retorna el puntero a la raíz del árbol rojo-negro.

```
CRedBlackTreeNode<T>* Root();
```

Valor devuelto

Retorna el puntero a la raíz.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si el árbol rojo-negro contiene el elemento con el valor indicado.

```
bool Contains(
   T  item     // valor buscado
   );
```

Parámetros

item

[in] Valor buscado.

Valor devuelto

Retorna true, si en el árbol rojo-negro existe el elemento con el valor indicado, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

Comparer

Retorna el puntero a la interfaz IComparer<T>, usada para organizar el árbol rojo-negro.

```
IComparer<T>* Comparer() const;
```

Valor devuelto

Retorna el puntero a la interfaz IComparer<T>.

© 2000-2025, MetaQuotes Ltd.

TryGetMin

Obtiene el elemento mínimo del árbol rojo-negro.

```
bool TryGetMin(
   T&  min     // variable para guardar el valor
   );
```

Parámetros

&min

[out] Variable en la que se guardará el valor mínimo.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

TryGetMax

Obtiene el elemento máximo del árbol rojo-negro.

```
bool TryGetMax(
   T&  max     // variable para guardar el valor
   );
```

Parámetros

&max

[out] Variable en la que se guardará el valor máximo.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todos los elementos del árbol rojo-negro a la matriz especificada, comenzando por un índice en particular.

```
int CopyTo(
   T&          dst_array[],     // matriz para el guardado
   const int   dst_start=0      // índice inicial para el guardado
   );
```

Parámetros

&dst_array[]

[out] Matriz en la que se guardarán los elementos del árbol rojo-negro.

dst_start=0

[in] Índice en la matriz desde el que se comienza el copiado.

Valor devuelto

Retorna el número de elementos copiados.

© 2000-2025, MetaQuotes Ltd.

Clear

Elimina todos los elementos del árbol rojo-negro.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Elimina la aparición del elemento indicado del árbol rojo-negro.

Versión para eliminar un elemento según el valor establecido.

```
bool Remove(
   T  value                        // valor del elemento
   );
```

Versión para eliminar un elemento según el puntero a un nodo.

```
bool Remove(
   CRedBlackTreeNode<T>*  node     // nodo del elemento
   );
```

Parámetros

item

[in] Valor del elemento que se debe eliminar.

**node*

[in] Nodo del elemento que se debe eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

RemoveMin

Elimina todos los elementos con el valor mínimo del árbol rojo-negro.

```
bool RemoveMin();
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

RemoveMax

Elimina todos los elementos con el valor máximo del árbol rojo-negro.

```
bool RemoveMax();
```

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Find

Busca la aparición del elemento establecido en el árbol rojo-negro.

```
CRedBlackTreeNode<T>* Find(
   T  value     // valor buscado
   );
```

Parámetros

value

[in] Valor buscado.

Valor devuelto

Retorna el puntero al nodo localizado que contenga el valor buscado en el caso de éxito, de lo contrario, NULL.

© 2000-2025, MetaQuotes Ltd.

FindMin

Busca el elemento con el valor mínimo del árbol rojo-negro.

```
CRedBlackTreeNode<T>* FindMin();
```

Valor devuelto

Retorna el puntero al nodo que contenga el valor mínimo en el caso de éxito, de lo contrario, NULL.

© 2000-2025, MetaQuotes Ltd.

FindMax

Busca el elemento con el valor máximo del árbol rojo-negro.

```
CRedBlackTreeNode<T>* FindMax();
```

Valor devuelto

Retorna el puntero al nodo que contenga el valor máximo en el caso de éxito, de lo contrario, NULL.

© 2000-2025, MetaQuotes Ltd.

CSortedMap<TKey,TValue>

La clase CSortedMap<TKey,TValue> es una clase genérica que implementa la interfaz IMap<TKey,TValue>.

Descripción

La clase CSortedMap<TKey,TValue> es la implementación de un recuadro hash dinámico cuyos datos se guardan en forma de parejas "clave — valor", clasificadas según la clave y respetando la necesidad de unicidad de la clave. Esta clase proporciona los métodos esenciales para trabajar con el recuadro hash: acceso al valor según la clave, búsqueda y eliminación de una pareja "clave-valor" y otros.

Declaración

```
template<typename TKey, typename TValue>
class CSortedMap : public IMap<TKey, TValue>
```

Encabezamiento

```
#include <Generic\SortedMap.mqh>
```

Jerarquía de herencia

ICollection

IMap

CSortedMap

Métodos de clase

Método	Descripción
Add	Añade una pareja "clave — valor" a un recuadro hash
Count	Retorna el número de elementos en el recuadro hash clasificado
Contains	Determina si el recuadro hash clasificado contiene la pareja "clave — valor" indicada
ContainsKey	Determina si contiene el recuadro hash clasificado la pareja "clave — valor" con la clave indicada
ContainsValue	Determina si el recuadro hash clasificado contiene la pareja "clave — valor" con el valor indicado
Comparer	Retorna el puntero a la interfaz IComparer<T>, usada para organizar el recuadro hash clasificado
CopyTo	Copia todas las parejas "clave — valor" de un recuadro hash clasificado a las matrices indicadas, comenzando por un índice determinado
Clear	Elimina todos los elementos de un recuadro hash clasificado
Remove	Elimina la primera aparición de una pareja "clave — valor" de un recuadro hash clasificado

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
TryGetValue	Obtiene un elemento del recuadro hash clasificado según la clave indicada
TrySetValue	Cambia el valor de una pareja "clave — valor" de un recuadro hash clasificado según la clave indicada

© 2000-2025, MetaQuotes Ltd.

Add

Añade una pareja "clave — valor" al recuadro hash.

Versión para añadir la pareja "clave — valor" formada.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // pareja "clave — valor"
   );
```

Versión para añadir una nueva pareja "clave — valor" con la clave y valor indicados.

```
bool Add(
   TKey    key,                         // clave
   TValue  value                        // valor
   );
```

Parámetros

**pair*

[in] Pareja "clave — valor".

key

[in] Clave.

value

[in] Valor.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Count

Retorna el número de elementos en el recuadro hash clasificado.

```
int Count();
```

© 2000-2025, MetaQuotes Ltd.

Comparer

Retorna el puntero a la interfaz IComparer<T>, usada para organizar el recuadro hash clasificado.

```
IComparer<TKey>* Comparer() const;
```

Valor devuelto

Retorna el puntero a la interfaz IComparer<T>.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si el recuadro hash clasificado contiene la pareja "clave — valor" indicada.

Versión para trabajar con la pareja "clave — valor" formada.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // pareja "clave — valor"
   );
```

Versión para trabajar con una pareja "clave — valor" en forma de clave y valor establecidos por separado.

```
bool Contains(
   TKey    key,                         // clave
   TValue  value                        // valor
   );
```

Parámetros

**item*

[in] Pareja "clave — valor".

key

[in] Clave.

value

[in] Valor.

Valor devuelto

Retorna true, si en el recuadro hash clasificado existe la pareja "clave-valor" con la clave y el valor establecidos, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ContainsKey

Determina si el recuadro hash clasificado contiene la pareja "clave — valor" con la clave indicada.

```
bool ContainsKey(
   TKey  key     // clave
   );
```

Parámetros

key

[in] Clave.

Valor devuelto

Retorna true, si en el recuadro hash clasificado existe la pareja "clave-valor" con la clave establecida, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ContainsValue

Determina si el recuadro hash clasificado contiene la pareja "clave — valor" con el valor indicado.

```
bool ContainsValue(
   TValue  value     // valor
   );
```

Parámetros

value

[in] Valor.

Valor devuelto

Retorna true, si en el recuadro hash clasificado existe la pareja "clave-valor" con el valor establecido, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todas las parejas "clave — valor" de un recuadro hash clasificado a las matrices indicadas, comenzando por un índice determinado.

Versión para copiar un recuadro hash clasificado a una matriz de parejas "clave — valor".

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // matriz para guardar las parejas "
   const int                    dst_start=0      // índice inicial para el guardado
   );
```

Versión para copiar un recuadro clasificado hash a matrices aparte para claves y valores.

```
int CopyTo(
   TKey&       dst_keys[],                       // matriz para guardar las claves
   TValue&     dst_values[],                     // matriz para guardar los valores
   const int   dst_start=0                       // índice inicial para el guardado
   );
```

Parámetros

**&dst_array[]*

[out] Matriz en la que se guardarán todas las parejas del recuadro hash.

&dst_keys[]

[out] Matriz en la que se guardarán todas las claves del recuadro hash.

&dst_values[]

[out] Matriz en la que se guardarán todos los valores del recuadro hash.

dst_start=0

[in] Índice en las matrices desde el que se comienza el copiado.

Valor devuelto

Retorna el número de parejas "clave - valor" copiadas.

© 2000-2025, MetaQuotes Ltd.

Clear

Elimina todos los elementos de un recuadro hash clasificado.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Elimina la primera aparición de una pareja "clave — valor" de un recuadro hash clasificado.

Versión para eliminar una pareja "clave — valor" según la pareja "clave — valor" formada.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // pareja "clave — valor"
   );
```

Versión para eliminar una pareja "clave — valor" según la clave.

```
bool Remove(
   TKey  key                            // clave
   );
```

Parámetros

**item*

[in] Pareja "clave — valor".

key

[in] Clave.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

TryGetValue

Obtiene un elemento del recuadro hash clasificado según la clave indicada.

```
bool TryGetValue(
   TKey      key,       // clave
   TValue&   value      // variable para guardar el valor
   );
```

Parámetros

key

[in] Clave.

&value

[out] Variable en la que se guardará el valor indicado de la pareja "clave — valor".

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

Cambia el valor de una pareja "clave — valor" de un recuadro hash según la clave indicada.

```
bool TrySetValue(
   TKey    key,       // clave
   TValue  value      // nuevo valor
   );
```

Parámetros

key

[in] Clave.

value

[in] Nuevo valor que se deber asignar a la pareja "clave — valor" indicada.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

CSortedSet<T>

La clase CSortedSet<T> es una clase genérica que implementa la interfaz ISet<T>.

Descripción

La clase CSortedSet<T> es la implementación de un conjunto dinámico clasificado de datos del tipo T, respetando las exigencias de unicidad de cada valor. Esta clase proporciona los métodos principales para trabajar con conjuntos, así como las operaciones con ellos: unificación y enumeración de conjuntos, definición de subjuntos rigurosos y no rigurosos y otros.

Declaración

```
template<typename T>
class CSortedSet : public ISet<T>
```

Encabezamiento

```
#include <Generic\SortedSet.mqh>
```

Jerarquía de herencia

ICollection
  ISet
    CSortedSet

Métodos de clase

Método	Descripción
Add	Añade un elemento al conjunto clasificado
Count	Retorna el número de elementos en el conjunto clasificado
Contains	Determina si el conjunto clasificado contiene el elemento con el valor indicado
Comparer	Retorna el puntero a la interfaz IComparer<T>, usada para organizar el conjunto clasificado
TryGetMin	Obtiene el elemento mínimo del conjunto clasificado
TryGetMax	Obtiene el elemento máximo del conjunto clasificado
CopyTo	Copia todos los elementos de un conjunto a la matriz especificada, comenzando por un índice en particular
Clear	Elimina todos los elementos de un conjunto clasificado
Remove	Elimina la aparición de el elemento indicado de un conjunto clasificado
ExceptWith	Realiza la operación de diferencia de la colección (matriz) actual y la transmitida

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
IntersectWith	Realiza la operación de intersección de la colección (matriz) actual y la transmitida
SymmetricExceptWith	Realiza la operación de diferencia simétrica de la colección (matriz) actual y la transmitida
UnionWith	Realiza la operación de unión de la colección (matriz) actual y la transmitida
IsProperSubsetOf	Determina si el conjunto clasificado actual es un subconjunto estricto de la colección o matriz establecidas
IsProperSupersetOf	Determina si el conjunto clasificado actual es un superconjunto estricto de la colección o matriz establecidas
IsSubsetOf	Determina si el conjunto clasificado actual es un subconjunto de la colección o matriz establecidas
IsSupersetOf	Determina si el conjunto clasificado actual es un superconjunto de la colección o matriz establecidas
Overlaps	Determina si el conjunto clasificado actual se cruza con la colección o matriz establecidas
SetEquals	Determina si el conjunto clasificado actual contiene todos los elementos de la colección o matriz establecidas
GetViewBetween	Obtiene del conjunto clasificado actual un subconjunto establecido por un valor mínimo y máximo
GetReverse	Obtiene una copia del conjunto clasificado actual, donde todos los elementos se ubican en el orden inverso

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento al conjunto clasificado.

```
bool Add(
   T  value     // valor del elemento
   );
```

Parámetros

value

[in] Valor del elemento a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Count

Retorna el número de elementos en el conjunto clasificado.

```
int Count();
```

Valor devuelto

Retorna el número de elementos.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si el conjunto clasificado contiene el elemento con el valor indicado.

```
bool Contains(
   T  item     // valor buscado
   );
```

Parámetros

item

[in] Valor buscado.

Valor devuelto

Retorna true, si en el conjunto existe un elemento con el valor indicado, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

Comparer

Retorna el puntero a la interfaz IComparer<T>, usada para organizar el conjunto clasificado.

```
IComparer<T>* Comparer() const;
```

Valor devuelto

Retorna el puntero a la interfaz IComparer<T>.

© 2000-2025, MetaQuotes Ltd.

TryGetMin

Obtiene el elemento mínimo del conjunto clasificado.

```
bool TryGetMin(
   T&  min     // variable para guardar el valor
   );
```

Parámetros

&min

[out] Variable en la que se guardará el valor mínimo.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

TryGetMax

Obtiene el elemento máximo del conjunto clasificado.

```
bool TryGetMax(
   T&  max     // variable para guardar el valor
   );
```

Parámetros

&max

[out] Variable en la que se guardará el valor máximo.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todos los elementos de un conjunto a la matriz especificada, comenzando por un índice en particular.

```
int CopyTo(
   T&          dst_array[],     // matriz para el guardado
   const int   dst_start=0      // índice inicial para el guardado
   );
```

Parámetros

&dst_array[]

[out] Matriz en la que se guardarán los elementos del conjunto.

dst_start=0

[in] Índice en la matriz desde el que se comienza el copiado.

Valor devuelto

Retorna el número de elementos copiados.

© 2000-2025, MetaQuotes Ltd.

Clear

Elimina todos los elementos de un conjunto clasificado.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Elimina la aparición del elemento indicado de un conjunto clasificado.

```
bool Remove(
   T  item     // valor del elemento
   );
```

Parámetros

item

[in] Valor del elemento que se debe eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

ExceptWith

Realiza la operación de diferencia de la colección (matriz) actual y la transmitida. Es decir, elimina de la colección (matriz) actual todos los elementos que también existen en la colección (matriz) indicada.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void ExceptWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección que se sustraerá del conjunto clasificado actual.

&collection[]

[in] Matriz que se sustraerá del conjunto clasificado actual.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

IntersectWith

Realiza la operación de intersección de la colección (matriz) actual y la transmitida. Es decir, deja en la colección (matriz) actual solo los elementos que también existen en la colección (matriz) indicada.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void IntersectWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección con la que se construirá la multiplicación.

&collection[]

[in] Matriz con la que se construirá la multiplicación.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

Realiza la operación de diferencia simétrica de la colección (matriz) actual y la transmitida. Es decir, en la colección (matriz) actual solo se contendrán aquellos elementos que existían o bien en el objeto fuente, o bien en el transmitido, pero no simultáneamente en ambos.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void SymmetricExceptWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección con la que se construirá la diferencia simétrica.

&collection[]

[in] Matriz con la que se construirá la diferencia simétrica.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

UnionWith

Realiza la operación de unión de la colección (matriz) actual y la transmitida. Es decir, añade a la colección (matriz) actual los elementos ausentes desde la colección (matriz) indicada.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // colección
   );
```

Versión para trabajar con la matriz.

```
void UnionWith(
   T&  array[]                     // matriz
   );
```

Parámetros

**collection*

[in] Colección con la que se construirá la suma.

&collection[]

[in] Matriz con la que se construirá la suma.

Nota

El resultado se guarda en la colección (matriz) actual.

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

Determina si el conjunto clasificado actual es un subconjunto estricto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsProperSubsetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto clasificado actual es un subconjunto estricto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsProperSupersetOf

Determina si el conjunto clasificado actual es un superconjunto estricto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsProperSupersetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto clasificado actual es un superconjunto estricto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsSubsetOf

Determina si el conjunto clasificado actual es un subconjunto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsSubsetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto clasificado actual es un subconjunto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

Determina si el conjunto clasificado actual es un superconjunto de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // colección para definir la relación
   );
```

Versión para trabajar con la matriz.

```
bool IsSupersetOf(
   T&  array[]                     // matriz para definir la relación
   );
```

Parámetros

**collection*

[in] Colección para definir la relación.

&collection[]

[in] Matriz para definir la relación.

Valor devuelto

Retorna true si el conjunto clasificado actual es un superconjunto, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

Overlaps

Determina si el conjunto clasificado actual se cruza con la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // colección para la comparación
   );
```

Versión para trabajar con la matriz.

```
bool Overlaps(
   T&  array[]                     // matriz para la comparación
   );
```

Parámetros

**collection*

[in] Colección para definir la intersección.

&collection[]

[in] Matriz para definir la intersección.

Valor devuelto

Retorna true si entre el conjunto clasificado actual y la colección o matriz existe una intersección, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SetEquals

Determina si el conjunto clasificado actual contiene todos los elementos de la colección o matriz establecidas.

Versión para trabajar con la colección que implementa la interfaz ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // colección para la comparación
   );
```

Versión para trabajar con la matriz.

```
bool SetEquals(
   T&  array[]                     // matriz para la comparación
   );
```

Parámetros

**collection*

[in] Colección para la comparación de elementos.

&collection[]

[in] Matriz para la comparación de elementos.

Valor devuelto

Retorna true si en el conjunto clasificado actual existen todos los elementos de la colección o matriz establecida, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

GetViewBetween

Obtiene del conjunto clasificado actual un subconjunto establecido por un valor mínimo y máximo.

```
bool GetViewBetween(
   T&  array[],          // matriz para el guardado
   T   lower_value,      // valor mínimo
   T   upper_value       // valor máximo
   );
```

Parámetros

&array[]

[out] Matriz para el guardado del subconjunto.

lower_value

[in] Valor mínimo del rango.

upper_value

[in] Valor máximo del rango.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

GetReverse

Obtiene una copia del conjunto clasificado actual, donde todos los elementos se ubican en el orden inverso.

```
bool GetReverse(
   T&  array[]     // matriz para el guardado
   );
```

Parámetros

&array[]

[out] Matriz para el guardado.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

CStack<T>

La clase CStack<T> es una clase genérica que implementa la interfaz ICollection<T>.

Descripción

La clase CStack<T> es una colección dinámica de datos del tipo T, organizada en forma de pila que funciona según el principio "última entrada — primera salida" (LIFO).

Declaración

```
template<typename T>
class CStack : public ICollection<T>
```

Encabezamiento

```
#include <Generic\Stack.mqh>
```

Jerarquía de herencia

ICollection

CStack

Métodos de clase

Método	Descripción
Add	Añade un elemento a la pila
Count	Retorna el número de elementos en la pila
Contains	Determina si la pila contiene el elemento con el valor indicado
TrimExcess	Reduce la capacidad de la pila hasta un número real de elementos
CopyTo	Copia todos los elementos de la pila a la matriz especificada, comenzando por un índice en particular
Clear	Elimina todos los elementos de la pila
Remove	Elimina la primera aparición del elemento especificado de la pila
Push	Añade un elemento a la pila
Peek	Retorna el elemento inicial, sin eliminarlo de la pila
Pop	Retorna el elemento inicial, eliminándolo de la pila

© 2000-2025, MetaQuotes Ltd.

Add

Añade un elemento a la pila.

```
bool Add(
   T  value     // valor del elemento
   );
```

Parámetros

value

[in] Valor del elemento a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Count

Retorna el número de elementos en la pila.

```
int Count();
```

Valor devuelto

Retorna el número de elementos.

© 2000-2025, MetaQuotes Ltd.

Contains

Determina si la pila contiene el elemento con el valor indicado.

```
bool Contains(
   T  item     // valor buscado
   );
```

Parámetros

item

[in] Valor buscado.

Valor devuelto

Retorna true, si en la pila existe un elemento con el valor indicado, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

Reduce la capacidad de un conjunto hasta un número real de elementos, liberando de esta forma la memoria no usada.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

Copia todos los elementos de la pila a la matriz especificada, comenzando por un índice en particular.

```
int CopyTo(
   T&          dst_array[],     // matriz para el guardado
   const int   dst_start=0      // índice inicial para el guardado
   );
```

Parámetros

&dst_array[]

[out] Matriz en la que se guardarán los elementos de la pila.

dst_start=0

[in] Índice en la matriz desde el que se comienza el copiado.

Valor devuelto

Retorna el número de elementos copiados.

© 2000-2025, MetaQuotes Ltd.

Clear

Elimina todos los elementos de la pila.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Elimina la primera aparición del elemento especificado de la pila.

```
bool Remove(
   T  item     // valor del elemento
   );
```

Parámetros

item

[in] Valor del elemento que se debe eliminar.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Push

Añade un elemento a la pila.

```
bool Push(
   T  value     // elemento a añadir
   );
```

Parámetros

value

[in] Elemento a añadir.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Peek

Retorna el elemento inicial, sin eliminarlo de la pila.

```
T Peek();
```

Valor devuelto

Retorna el elemento inicial.

© 2000-2025, MetaQuotes Ltd.

Pop

Retorna el elemento inicial, eliminándolo de la pila.

```
T Pop();
```

Valor devuelto

Retorna el elemento inicial.

© 2000-2025, MetaQuotes Ltd.

ArrayBinarySearch

Busca el valor indicado en una matriz unidimensional clasificada en orden ascendente, usando la interfaz IComparable<T> para comparar los elementos.

```
template<typename T>
int ArrayBinarySearch(
   T&              array[],         // matriz para la búsqueda
   const int       start_index,     // índice inicial
   const int       count,           // rango de búsqueda
   T               value,           // valor buscado
   IComparer<T>*   comparer         // interfaz para la comparación
   );
```

Parámetros

&array[]

[out] Matriz para la búsqueda.

value

[in] Valor buscado.

**comparer*

[in] Interfaz para comparar los elementos.

start_index

[in] Índice inicial desde el que comienza la búsqueda.

count

[in] Longitud del rango de búsqueda.

Valor devuelto

Retorna el índice del elemento encontrado. Si el valor buscado no se localiza, retorna el índice del menor elemento que se encuentre más cerca según su valor.

© 2000-2025, MetaQuotes Ltd.

ArrayIndexOf

Busca la primera aparición de un valor en una matriz unidimensional.

```
template<typename T>
int ArrayIndexOf(
   T&          array[],          // matriz para la búsqueda
   T           value,            // valor buscado
   const int   start_index,      // índice inicial
   const int   count             // rango de búsqueda
   );
```

Parámetros

&array[]

[out] Matriz para la búsqueda.

value

[in] Valor buscado.

start_index

[in] Índice inicial desde el que comienza la búsqueda.

count

[in] Longitud del rango de búsqueda.

Valor devuelto

Retorna el índice del primer elemento encontrado. Si no se localiza el valor, retorna -1.

© 2000-2025, MetaQuotes Ltd.

ArrayLastIndexOf

Busca la última aparición de un valor en una matriz unidimensional.

```
template<typename T>
int ArrayLastIndexOf(
   T&          array[],          // matriz para la búsqueda
   T           value,            // valor buscado
   const int   start_index,      // índice inicial
   const int   count             // rango de búsqueda
   );
```

Parámetros

&array[]

[out] Matriz para la búsqueda.

value

[in] Valor buscado.

start_index

[in] Índice inicial desde el que comienza la búsqueda.

count

[in] Longitud del rango de búsqueda.

Valor devuelto

Retorna el índice del último elemento encontrado. Si no se localiza el valor, retorna -1.

© 2000-2025, MetaQuotes Ltd.

ArrayReverse

Cambia la secuenciación de los elementos en una matriz unidimensional.

```
template<typename T>
bool ArrayReverse(
   T&          array[],          // matriz fuente
   const int   start_index,      // índice inicial
   const int   count             // número de elementos
   );
```

Parámetros

&array[]

[out] Matriz fuente.

start_index

[in] Índice inicial.

count

[in] Número de elementos de la matriz que participan en la operación.

Valor devuelto

Devuelve true en caso de éxito, de lo contrario, devuelve false.

© 2000-2025, MetaQuotes Ltd.

Compare

Compara dos valores sobre la relación "Mayor, menor o igual".

Versión para comparar dos valores del tipo bool.

```
int Compare(
   const bool  x,          // primer valor
   const bool  y           // segundo valor
   );
```

Versión para comparar dos valores del tipo char.

```
int Compare(
   const char  x,          // primer valor
   const char  y           // segundo valor
   );
```

Versión para comparar dos valores del tipo uchar.

```
int Compare(
   const uchar  x,         // primer valor
   const uchar  y          // segundo valor
   );
```

Versión para comparar dos valores del tipo short.

```
int Compare(
   const short  x,         // primer valor
   const short  y          // segundo valor
   );
```

Versión para comparar dos valores del tipo ushort.

```
int Compare(
   const ushort  x,        // primer valor
   const ushort  y         // segundo valor
   );
```

Versión para comparar dos valores del tipo color.

```
int Compare(
   const color  x,         // primer valor
   const color  y          // segundo valor
   );
```

Versión para comparar dos valores del tipo int.

```
int Compare(
   const int  x,           // primer valor
   const int  y            // segundo valor
   );
```

© 2000-2025, MetaQuotes Ltd.

Versión para comparar dos valores del tipo uint.

```
int Compare(
   const uint  x,          // primer valor
   const uint  y           // segundo valor
   );
```

Versión para comparar dos valores del tipo datetime.

```
int Compare(
   const datetime  x,      // primer valor
   const datetime  y       // segundo valor
   );
```

Versión para comparar dos valores del tipo long.

```
int Compare(
   const long  x,          // primer valor
   const long  y           // segundo valor
   );
```

Versión para comparar dos valores del tipo ulong.

```
int Compare(
   const ulong  x,         // primer valor
   const ulong  y          // segundo valor
   );
```

Versión para comparar dos valores del tipo float.

```
int Compare(
   const float  x,         // primer valor
   const float  y          // segundo valor
   );
```

Versión para comparar dos valores del tipo double.

```
int Compare(
   const double  x,        // primer valor
   const double  y         // segundo valor
   );
```

Versión para comparar dos valores del tipo string.

```
int Compare(
   const string  x,        // primer valor
   const string  y         // segundo valor
   );
```

Versión para comparar dos valores del resto de tipos.

```
template<typename T>
int Compare(
```

© 2000-2025, MetaQuotes Ltd.

```
   T  x,                  // primer valor
   T  y                   // segundo valor
   );
```

Parámetros

x

[in] Primer valor.

y

[in] Segundo valor

Valor devuelto

Retorna un número que expresa la relación de los dos valores comparados:

- resultado menor a cero — x menor a y (x<y)
- resultado igual a cero — x igual a y (x=y)
- resultado mayor a cero — x mayor a y (x>y)

Nota

Si el tipo es T, el objeto que implementa la interfaz IComparable<T>, los objetos se compararán usando como base su método de comparación Compare. En el resto de los casos se retorna 0.

© 2000-2025, MetaQuotes Ltd.

Equals

Compara dos valores sobre su igualdad.

```
template<typename T>
bool Equals(
   T  x,      // primer valor
   T  y       // segundo valor
   );
```

Parámetros

x

[in] Primer valor.

y

[in] Segundo valor.

Valor devuelto

Retorna true si los objetos son iguales, de lo contrario, false.

Nota

Si el tipo es T, el objeto que implementa la interfaz IEqualityComparable<T>, los objetos se compararán usando como base su método de comparación Equals. En el resto de los casos se aplicará la comparación estándar sobre la igualdad.

© 2000-2025, MetaQuotes Ltd.

GetHashCode

Calcula el valor del código hash.

Versión para trabajar con el tipo bool.

```
int GetHashCode(
   const bool   value          // valor
   );
```

Versión para trabajar con el tipo char.

```
int GetHashCode(
   const char   value          // valor
   );
```

Versión para trabajar con el tipo uchar.

```
int GetHashCode(
   const uchar   value         // valor
   );
```

Versión para trabajar con el tipo short.

```
int GetHashCode(
   const short   value         // valor
   );
```

Versión para trabajar con el tipo ushort.

```
int GetHashCode(
   const ushort   value        // valor
   );
```

Versión para trabajar con el tipo color.

```
int GetHashCode(
   const color   value         // valor
   );
```

Versión para trabajar con el tipo int.

```
int GetHashCode(
   const int   value           // valor
   );
```

Versión para trabajar con el tipo uint.

```
int GetHashCode(
   const uint   value          // valor
   );
```

Versión para trabajar con el tipo datetime.

```
int GetHashCode(
```

© 2000-2025, MetaQuotes Ltd.

```
   const datetime   value     // valor
   );
```

Versión para trabajar con el tipo long.

```
int GetHashCode(
   const long   value         // valor
   );
```

Versión para trabajar con el tipo ulong.

```
int GetHashCode(
   const ulong   value        // valor
   );
```

Versión para trabajar con el tipo float.

```
int GetHashCode(
   const float   value        // valor
   );
```

Versión para trabajar con el tipo double.

```
int GetHashCode(
   const double   value       // valor
   );
```

Versión para trabajar con el tipo string.

```
int GetHashCode(
   const string   value       // valor
   );
```

Versión para trabajar con el resto de los tipos.

```
template<typename T>
int GetHashCode(
   T   value                  // valor
   );
```

Parámetros

value

[in] Valor para el que se debe obtener el código hash.

Valor devuelto

Retorna el código hash.

Nota

Si el tipo es T, el objeto que implementa la interfaz IEqualityComparable<T>, el código hash se obtendrá usando como base su método HashCode. En el resto de los casos, el código hash se calculará como el valor del código hash a partir del nombre del tipo de ese valor.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Operaciones con archivos

Esta sección contiene detalles técnicos del funcionamiento de las clases que implementan las operaciones con archivos, así como las descripciones correspondientes de los componentes de la Librería Estándar del lenguaje MQL5.

Las clases de operaciones de archivos le ayudarán a ahorrar tiempo en el momento de desarrollar sus propias aplicaciones que realizan operaciones de entrada y salida con ficheros.

La Librería Estándar de MQL5 se encuentra en el directorio de trabajo del terminal Include\Files.

Class	Descripción
CFile	Clase base de las operaciones con archivos
CFileBin	Clase de las operaciones con archivos binarios
CFileTxt	Clase de las operaciones con archivos de texto

© 2000-2025, MetaQuotes Ltd.

CFile

La clase CFile es la clase base de las clases CFileBin y CFileTxt.

Descripción

La clase CFile facilita el acceso a las funciones de archivo del API MQL5, así como a las funciones de carpetas.

Declaración

```
class CFile: public CObject
```

Título

```
#include <Files\File.mqh>
```

Jerarquía de herencia

CObject

CFile

Descendientes directos

CFileBin, CFilePipe, CFileTxt

Métodos de la clase

Atributos	
Handle	Obtiene el manejador del archivo
Filename	Obtiene el nombre del archivo
Flags	Obtiene la bandera del archivo
SetUnicode	Establece/Reinicia la bandera FILE_UNICODE
SetCommon	Establece/Reinicia la bandera FILE_COMMON
Métodos generales de archivo	
Open	Abre el archivo
Close	Cierra el archivo
Delete	Borra el archivo
IsExist	Comprueba si el archivo existe
Copy	Copia el archivo
Move	Cambia el nombre/mueve el archivo
Size	Obtiene el tamaño del archivo

© 2000-2025, MetaQuotes Ltd.

Atributos	
Tell	Obtiene la posición actual del archivo
Seek	Establece la posición actual del archivo
Flush	Vuelca los datos al disco
IsEnding	Comprueba el final del archivo
IsLineEnding	Comprueba el final de la línea
Métodos generales de las carpetas	
FolderCreate	Crea la carpeta
FolderDelete	Borra la carpeta
FolderClean	Libera la carpeta
Métodos de búsqueda	
FileFindFirst	Comienza la búsqueda del archivo
FileFindNext	Continúa la búsqueda del archivo
FileFindClose	Cierra el manejador de la búsqueda

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Handle

Obtiene el manejador del archivo abierto.

```
int  Handle()
```

Valor devuelto

Manejador del archivo abierto, asignado a la instancia de la clase. Si no hay ningún archivo asignado, devuelve -1.

© 2000-2025, MetaQuotes Ltd.

FileName

Obtiene el nombre del archivo abierto.

```
string  FileName()
```

Valor devuelto

Nombre del archivo abierto asignado a la instancia de la clase. Si no hay ningún archivo asignado, devuelve "".

© 2000-2025, MetaQuotes Ltd.

Flags

Obtiene las banderas del archivo abierto.

```
int  Flags()
```

Valor devuelto

Banderas del archivo abierto, asignado a la instancia de la clase.

© 2000-2025, MetaQuotes Ltd.

SetUnicode

Establece/Libera la bandera FILE_UNICODE.

```
void  SetUnicode(
   bool  unicode      // Valor nuevo de la bandera
   )
```

Parámetros

unicode

[in] Valor nuevo de la bandera FILE_UNICODE.

Nota

El resultado de las operaciones string depende de la bandera FILE_UNICODE. Si es false, se utilizan los códigos ANSI (símbolos de un byte). Si está establecido, se utilizan los códigos UNICODE (símbolos de dos bytes). Si el archivo ya está abierto, la bandera no se puede cambiar.

© 2000-2025, MetaQuotes Ltd.

SetCommon

Establece/Libera la bandera FILE_COMMON.

```
void  SetCommon(
   bool  common      // Valor nuevo de la bandera
   )
```

Parámetros

common

[in] Nuevo valor de la bandera FILE_COMMON.

Nota

La bandera FILE_UNICODE determina la carpeta de trabajo actual. Si es false, se utiliza la carpeta local del terminal como carpeta de trabajo actual. Si es true, se utiliza la carpeta general como carpeta de trabajo actual. Si el archivo ya está abierto, la bandera no se puede cambiar.

© 2000-2025, MetaQuotes Ltd.

Open

Abre el archivo especificado y en caso de éxito lo asigna a la instancia de la clase.

```
int  Open(
   const string  file_name,        // Nombre del archivo
   int           flags,            // Banderas
   short         delimiter=9       // Separador
   )
```

Parámetros

file_name

[in] Nombre del archivo a abrir.

flags

[in] Banderas del archivo.

delimiter=9

[in] Separador del archivo CSV.

Valor devuelto

Manejador del archivo abierto.

Nota

La carpeta de trabajo depende de la bandera FILE_COMMON, definida en el método SetCommon().

© 2000-2025, MetaQuotes Ltd.

Close

Cierra el archivo, asignado a la instancia de la clase.

```
void  Close()
```

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el archivo, asignado a la instancia del archivo.

```
void  Delete()
```

Delete

Borra el archivo especificado.

```
void  Delete(
   const string  file_name      // Nombre del archivo
   )
```

Parámetros

file_name

[in] Nombre del archivo a borrar.

Nota

La carpeta de trabajo depende de la bandera FILE_COMMON, definida en el método SetCommon().

© 2000-2025, MetaQuotes Ltd.

IsExist

Comprueba si el archivo existe

```
bool  IsExist(
   const string  file_name      // Nombre del archivo
   )
```

Parámetros

file_name

[in] Nombre del archivo a comprobar.

Valor devuelto

true, si existe el archivo.

© 2000-2025, MetaQuotes Ltd.

Copy

Copia el archivo.

```
bool  Copy(
   const string  src_name,      // Nombre del archivo fuente
   int           src_flag,      // Bandera
   const string  dst_name,      // Nombre del archivo destino
   int           dst_flags      // Banderas
   )
```

Parámetros

src_name

[in] Nombre del archivo a copiar.

src_flag

[in] Banderas del archivo a copiar (solo se utiliza FILE_COMMON).

dst_name

[in] Nombre del archivo destino.

dst_flags

[in] Banderas del archivo destino (solo se utilizan FILE_REWRITE y FILE_COMMON).

Valor devuelto

true si se ejecuta correctamente, false si la copia no se lleva a cabo.

© 2000-2025, MetaQuotes Ltd.

Move

Cambia el nombre/mueve el archivo.

```
bool  Move(
   const string  src_name,      // Nombre del archivo fuente
   int           src_flag,      // Bandera
   const string  dst_name,      // Nombre del archivo destino
   int           dst_flags      // Banderas
   )
```

Parámetros

src_name

[in] Nombre del archivo a mover.

src_flag

[in] Banderas del archivo a copiar (solo se utiliza FILE_COMMON).

dst_name

[in] Nombre del archivo destino.

dst_flags

[in] Banderas del archivo destino (solo se utilizan FILE_REWRITE y FILE_COMMON).

Valor devuelto

true si se ejecuta correctamente, false si no se puede mover el archivo.

© 2000-2025, MetaQuotes Ltd.

Size

Obtiene el tamaño del archivo en bytes.

```
ulong  Size()
```

Valor devuelto

Tamaño del archivo en bytes. Si no se asigna ningún archivo, devuelve ULONG_MAX.

© 2000-2025, MetaQuotes Ltd.

Tell

Obtiene la posición actual del archivo.

```
ulong  Tell()
```

Valor devuelto

La posición actual del archivo. Si no se asigna ningún archivo, devuelve ULONG_MAX.

© 2000-2025, MetaQuotes Ltd.

Seek

Establece la posición actual del archivo.

```
void  Seek(
   long                offset,     // Offset
   ENUM_FILE_POSITION  origin      // Origen
   )
```

Parámetros

offset

[in] Offset del archivo en bytes (puede ser negativo).

origin

[in] Origen del offset.

Valor devuelto

true si se ejecuta correctamente, false si no se puede cambiar la posición del archivo.

© 2000-2025, MetaQuotes Ltd.

Flush

Vuelca en el disco los datos del búfer de entrada/salida del archivo.

```
void  Flush()
```

© 2000-2025, MetaQuotes Ltd.

IsEnding

Comprueba el final del archivo. Se utiliza en las operaciones de lectura del archivo.

```
bool  IsEnding()
```

Valor devuelto

true si se alcanza el final del archivo tras las operaciones de lectura o búsqueda.

© 2000-2025, MetaQuotes Ltd.

IsLineEnding

Comprueba el final de la línea. Se utiliza en las operaciones de lectura del archivo.

```
bool  IsLineEnding()
```

Valor devuelto

true si se alcanza el final de la línea tras la operación de lectura del archivo txt o csv (caracteres CR-LF).

© 2000-2025, MetaQuotes Ltd.

FolderCreate

Crea la nueva carpeta.

```
bool  FolderCreate(
   const string  folder_name      // Nombre de la carpeta
   )
```

Parámetros

folder_name

[in] Nombre de la carpeta a crear. Contiene la ruta relativa de la carpeta definida en la bandera FILE_COMMON.

Valor devuelto

true si se ejecuta correctamente, false si no se puede crear la carpeta.

Nota

La carpeta de trabajo depende de la bandera FILE_COMMON, definida en el método SetCommon().

© 2000-2025, MetaQuotes Ltd.

FolderDelete

Borra la carpeta especificada.

```
bool  FolderDelete(
   const string  folder_name      // Nombre de la carpeta
   )
```

Parámetros

folder_name

[in] Nombre de la carpeta a borrar. Contiene la ruta relativa de la carpeta definida en la bandera FILE_COMMON.

Valor devuelto

true si se ejecuta correctamente, false si la carpeta no se puede borrar.

Nota

La carpeta de trabajo depende de la bandera FILE_COMMON, definida en el método SetCommon().

© 2000-2025, MetaQuotes Ltd.

FolderClean

Limpia la carpeta especificada.

```
bool  FolderClean(
   const string  folder_name      // Nombre de la carpeta
   )
```

Parámetros

folder_name

[in] Nombre de la carpeta a borrar. Contiene la ruta relativa de la carpeta definida en la bandera FILE_COMMON.

Valor devuelto

true si se ejecuta correctamente, false si no se puede limpiar la carpeta.

Nota

La carpeta de trabajo depende de la bandera FILE_COMMON, definida en el método SetCommon().

© 2000-2025, MetaQuotes Ltd.

FileFindFirst

Comienza la búsqueda en el archivo de acuerdo al filtro especificado.

```
int  FileFindFirst(
   const string  filter,         // Filtro de búsqueda
   string&       file_name       // Referencia a la cadena
   )
```

Parámetros

filter

[in] Filtro de búsqueda.

file_name

[out] Referencia a la cadena del primer archivo encontrado.

Valor devuelto

Si se ejecuta correctamente devuelve el manejador, que puede utilizarse para realizar otra búsqueda en el archivo mediante el método FileFindNext; o INVALID_HANDLE, si el filtro especificado no se corresponde con ningún archivo.

Nota

La carpeta de trabajo depende de la bandera FILE_COMMON, definida en el método SetCommon().

© 2000-2025, MetaQuotes Ltd.

FileFindNext

Continúa la búsqueda comenzada por la función FileFindFirst().

```
bool  FileFindNext(
   int      search_handle,     // Manejador de la búsqueda
   string&  file_name          // Referencia a la cadena del siguiente archivo encontr
   )
```

Parámetros

search_handle

[in] Manejador de la búsqueda, devuelto por el método FileFindFirst().

file_name

[in] La referencia a la cadena del nombre del archivo encontrado, si se ejecuta correctamente.

Valor devuelto

true si se ejecuta correctamente; false si el filtro especificado no se corresponde con ningún archivo.

© 2000-2025, MetaQuotes Ltd.

FileFindClose

Cierra el manejador de la búsqueda.

```
void  FileFindClose(
   int  search_handle      // Manejador de la búsqueda
   )
```

Parámetros

search_handle

[in] Manejador de la búsqueda, devuelto por el método FileFindFirst().

© 2000-2025, MetaQuotes Ltd.

CFileBin

La clase CFileBin facilita el acceso a los archivos binarios.

Descripción

La clase CFileBin proporciona acceso a los archivos binarios.

Declaración

```
class CFileBin: public CFile
```

Título

```
#include <Files\FileBin.mqh>
```

Jerarquía de herencia

CObject
  CFile
    CFileBin

Métodos de la clase

Métodos de apertura	
Open	Abre un archivo binario
Métodos de escritura	
WriteChar	Escribe la variable de tipo char o uchar
WriteShort	Escribe la variable de tipo short o ushort
WriteInteger	Escribe la variable de tipo int o uint
WriteLong	Escribe la variable de tipo long o ulong
WriteFloat	Escribe la variable de tipo float
WriteDouble	Escribe la variable de tipo double
WriteString	Escribe la variable de tipo string
WriteCharArray	Escribe un array de variables de tipo char o uchar
WriteShortArray	Escribe un array de variables de tipo short o ushort
WriteIntegerArray	Escribe un array de variables de tipo int o uint
WriteLongArray	Escribe un array de variables de tipo long o ulong
WriteFloatArray	Escribe un array de variables float
WriteDoubleArray	Escribe un array de variables de tipo double
WriteObject	Escribe datos de la instancia de la clase CObject

© 2000-2025, MetaQuotes Ltd.

Métodos de apertura	
Métodos de lectura	
ReadChar	Lee la variable de tipo char o uchar
ReadShort	Lee la variable de tipo short o ushort
ReadInteger	Lee la variable de tipo int o uint
ReadLong	Lee la variable de tipo long o ulong
ReadFloat	Lee la variable de tipo float
ReadDouble	Lee la variable de tipo double
ReadString	Lee la variable de tipo string
ReadCharArray	Lee el array de variables de tipo char o uchar
ReadShortArray	Lee el array de variables de tipo short o ushort
ReadIntegerArray	Lee el array de variables de tipo int o uint
ReadLongArray	Lee el array de variables de tipo long o ulong
ReadFloatArray	Lee el array de variables de tipo float
ReadDoubleArray	Lee el array de variables de tipo double
ReadObject	Escribe datos de la instancia de la clase CObject

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CFile

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

© 2000-2025, MetaQuotes Ltd.

Open

Abre el archivo binario especificado y en caso de éxito lo asigna a la instancia de la clase.

```
int  Open(
   const string  file_name,      // Nombre del archivo
   int           flags           // Banderas
   )
```

Parámetros

file_name

[in] Nombre del archivo a abrir.

flags

[in] Banderas de apertura del archivo (hay un ajuste forzado de la bandera FILE_BIN).

Valor devuelto

Manejador del archivo abierto.

© 2000-2025, MetaQuotes Ltd.

WriteChar

Escribe en el archivo la variable de tipo char o uchar.

```
uint  WriteChar(
   char  value      // Valor
   )
```

Parámetros

value

[in] Variable a escribir.

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteShort

Escribe en el archivo la variable de tipo short o ushort.

```
uint  WriteShort(
   short  value      // Valor
   )
```

Parámetros

value

[in] Variable a escribir.

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteInteger

Escribe en el archivo la variable de tipo int o uint.

```
uint  WriteInteger(
   int  value      // Valor
   )
```

Parámetros

value

[in] Variable a escribir.

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteLong

Escribe en el archivo las variables de tipo long o ulong.

```
uint  WriteLong(
   long  value      // Valor
   )
```

Parámetros

value

[in] Variable a escribir.

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteFloat

Escribe en el archivo la variable de tipo float.

```
uint  WriteFloat(
   float  value      // Valor
   )
```

Parámetros

value

[in] Variable a escribir.

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteDouble

Escribe en el archivo la variable de tipo double.

```
uint  WriteDouble(
   double  value      // Valor
   )
```

Parámetros

value

[in] Variable a escribir.

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteString

Escribe en el archivo la variable de tipo string.

```
uint  WriteString(
   const string  value      // Valor
   )
```

Parámetros

value

[in] String a escribir.

Valor devuelto

Número de bytes escritos.

WriteString

Escribe en el archivo la variable de tipo string.

```
uint  WriteString(
   const string  value,     // Valor
   int           size       // Tamaño
   )
```

Parámetros

value

[in] String a escribir.

size

[in] Número de bytes a escribir.

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteCharArray

Escribe en el archivo el array de variables de tipo char o uchar.

```
uint  WriteCharArray(
   char&  array[],            // Referencia al array
   int    start_item=0,       // Elemento de inicio
   int    items_count=-1      // Número de elementos
   )
```

Parámetros

array[]

[in] Array a escribir.

start_item=0

[in] Elemento de inicio por donde empezar a escribir.

items_count=-1

[in] Número de elementos a escribir (-1 - para el array entero).

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteShortArray

Escribe en el archivo el array de variables de tipo short o ushort.

```
uint  WriteShortArray(
   short&  array[],            // Array a escribir
   int     start_item=0,       // Elemento de inicio
   int     items_count=-1      // Número de elementos a escribir
   )
```

Parámetros

array[]

[in] Array a escribir.

start_item=0

[in] Elemento de inicio por donde empezar a escribir.

items_count=-1

[in] Número de elementos a escribir (-1 - para el array entero).

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteIntegerArray

Escribe en el archivo el array de variables de tipo int o uint.

```
uint  WriteIntegerArray(
   int&  array[],            // Array a escribir
   int   start_item=0,       // Elemento de inicio
   int   items_count=-1      // Número de elementos a escribir
   )
```

Parámetros

array[]

[in] Array a escribir.

start_item=0

[in] Elemento de inicio por donde empezar a escribir.

items_count=-1

[in] Número de elementos a escribir (-1 - para el array entero).

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteLongArray

Escribe en el archivo el array de variables de tipo long o ulong.

```
uint  WriteLongArray(
   long&  array[],            // Array a escribir
   int    start_item=0,       // Elemento de inicio
   int    items_count=-1      // Número de elementos a escribir
   )
```

Parámetros

array[]

[in] Array a escribir.

start_item=0

[in] Elemento de inicio por donde empezar a escribir.

items_count=-1

[in] Número de elementos a escribir (-1 - para el array entero).

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteFloatArray

Escribe en el archivo el array de variables de tipo float.

```
uint  WriteFloatArray(
   float&  array[],            // Array a escribir
   int     start_item=0,       // Elemento de inicio
   int     items_count=-1      // Número de elementos a escribir
   )
```

Parámetros

array[]

[in] Array a escribir.

start_item=0

[in] Elemento de inicio por donde empezar a escribir.

items_count=-1

[in] Número de elementos a escribir (-1 - para el array entero).

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteDoubleArray

Escribe en el archivo el array de variables de tipo double.

```
uint  WriteDoubleArray(
   double&  array[],             // Array a escribir
   int      start_item=0,        // Elemento de inicio
   int      items_count=-1       // Número de elementos a escribir
   )
```

Parámetros

array[]

[in] Array a escribir.

start_item=0

[in] Elemento de inicio por donde empezar a escribir.

items_count=-1

[in] Número de elementos a escribir (-1 - para el array entero).

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

WriteObject

Escribe en el archivo los datos de la instancia de la clase CObject.

```
bool  WriteObject(
   CObject*  object      // Referencia al objeto
   )
```

Parámetros

object

[in] Referencia a la instancia de la clase CObject para escritura.

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden escribir.

© 2000-2025, MetaQuotes Ltd.

ReadChar

Lee del archivo la variable de tipo char o uchar.

```
bool  ReadChar(
   char&  value      // Variable objetivo
   )
```

Parámetros

value

[in] Variable objetivo de tipo char.

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadShort

Lee del archivo la variable de tipo short o ushort.

```
bool  ReadShort(
   short&  value
   )
```

Parámetros

value

[in] Variable objetivo de tipo short o ushort.

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadInteger

Lee del archivo la variable de tipo int o uint.

```
bool  ReadInteger(
   int&  value      // Variable objetivo
   )
```

Parámetros

value

[in] Variable objetivo de tipo int o uint.

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadLong

Lee del archivo la variable de tipo long o ulong.

```
bool  ReadLong(
   long&  value
   )
```

Parámetros

value

[in] Variable objetivo de tipo long o ulong.

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadFloat

Lee del archivo la variable de tipo float.

```
bool  ReadFloat(
   float&  value      // Variable objetivo
   )
```

Parámetros

value

[in] Variable objetivo de tipo float.

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadDouble

Lee del archivo la variable de tipo double.

```
bool  ReadDouble(
   double&  value
   )
```

Parámetros

value

[in] Variable objetivo de tipo double.

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadString

Lee del archivo la variable de tipo string.

```
bool  ReadString(
   string&  value      // String objetivo
   )
```

Parámetros

value

[in] Variable objetivo de tipo string.

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

ReadString

Lee del archivo la variable de tipo string.

```
bool  ReadString(
   string&  value
   )
```

Parámetros

value

[in] Variable objetivo de tipo string.

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadCharArray

Lee del archivo el array de variables de tipo char o uchar.

```
bool  ReadCharArray(
   char&  array[],            // Array objetivo
   int    start_item=0,       // Elemento de inicio
   int    items_count=-1      // Número de elementos a leer
   )
```

Parámetros

array[]

[in] Referencia al array objetivo de tipo char o uchar.

start_item=0

[in] Elemento de inicio por donde se empieza a leer.

items_count=-1

[in] Número de elementos a leer (-1 - leer al final del archivo).

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadShortArray

Lee del archivo el array de variables de tipo short o ushort.

```
bool  ReadShortArray(
   short&  array[],            // Array objetivo
   int     start_item=0,       // Elemento de inicio
   int     items_count=-1      // Número de elementos a leer
   )
```

Parámetros

array[]

[in] Referencia al array objetivo de tipo short o ushort.

start_item=0

[in] Elemento de inicio por donde se empieza a leer.

items_count=-1

[in] Número de elementos a leer (-1 - leer al final del archivo).

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadIntegerArray

Lee del archivo el array de variables de tipo int o uint.

```
bool  ReadIntegerArray(
   int&  array[],             // Array objetivo
   int   start_item=0,        // Elemento de inicio
   int   items_count=-1       // Número de elementos a leer
   )
```

Parámetros

array[]

[in] Referencia al array objetivo de tipo int o uint.

start_item=0

[in] Elemento de inicio por donde se empieza a leer.

items_count=-1

[in] Número de elementos a leer (-1 - leer al final del archivo).

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadLongArray

Lee del archivo el array de variables de tipo long o ulong.

```
bool  ReadLongArray(
   long&  array[],            // Array objetivo
   int    start_item=0,       // Elemento de inicio
   int    items_count=-1      // Número de elementos a leer
   )
```

Parámetros

array[]

[in] Referencia al array objetivo de tipo long o ulong.

start_item=0

[in] Elemento de inicio por donde se empieza a leer.

items_count=-1

[in] Número de elementos a leer (-1 - leer al final del archivo).

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadFloatArray

Lee del archivo el array de variables de tipo float.

```
bool  ReadFloatArray(
   float&  array[],            // Array objetivo
   int     start_item=0,       // Elemento de inicio
   int     items_count=-1      // Número de elementos a leer
   )
```

Parámetros

array[]

[in] Referencia al array objetivo de tipo float.

start_item=0

[in] Elemento de inicio por donde se empieza a leer.

items_count=-1

[in] Número de elementos a leer (-1 - leer al final del archivo).

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadDoubleArray

Lee del archivo el array de variables de tipo double.

```
bool  ReadDoubleArray(
   double&  array[],             // Array objetivo
   int      start_item=0,        // Elemento de inicio
   int      items_count=-1       // Número de elementos a leer
   )
```

Parámetros

array[]

[in] Referencia al array objetivo de tipo double.

start_item=0

[in] Elemento de inicio por donde se empieza a leer.

items_count=-1

[in] Número de elementos a leer (-1 - leer al final del archivo).

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

ReadObject

Lee del archivo los datos de la instancia de la clase CObject.

```
bool  ReadObject(
   CObject*  object      // Referencia al objeto
   )
```

Parámetros

object

[in] Referencia a la instancia de la clase CObject para lectura.

Valor devuelto

true si se ejecuta correctamente, false si los datos no se pueden leer.

© 2000-2025, MetaQuotes Ltd.

CFileTxt

La clase CFileTxt facilita el acceso a los archivos de texto.

Descripción

La clase CFileTxt proporciona acceso a los archivos de texto.

Declaración

```
class CFileTxt: public CFile
```

Título

```
#include <Files\FileTxt.mqh>
```

Jerarquía de herencia

CObject
  CFile
    CFileTxt

Métodos de la clase

Métodos de apertura	
Open	Abre el archivo de texto
Métodos de escritura	
WriteString	Escribe en el archivo la variable de tipo string
Métodos de lectura	
ReadString	Lee del archivo la variable de tipo string

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CFile

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

© 2000-2025, MetaQuotes Ltd.

Open

Abre el archivo de texto especificado y en caso de éxito lo asigna a la instancia de la clase.

```
int  Open(
   const string  file_name,     // nombre del archivo
   int           flags          // banderas
   )
```

Parámetros

file_name

[in] Nombre del archivo a abrir.

flags

[in] Banderas de apertura del archivo (hay un ajuste forzado de la bandera FILE_TXT).

Valor devuelto

Manejador del archivo abierto.

© 2000-2025, MetaQuotes Ltd.

WriteString

Escribe en el archivo la variable de tipo string.

```
uint  WriteString(
   const string  value      // String a escribir
   )
```

Parámetros

value

[in] String a escribir.

Valor devuelto

Número de bytes escritos.

© 2000-2025, MetaQuotes Ltd.

ReadString

Lee del archivo la variable de tipo string.

```
string  ReadString()
```

Valor devuelto

String que ha sido leída.

© 2000-2025, MetaQuotes Ltd.

Operaciones con cadenas de caracteres

Esta sección contiene detalles técnicos de las clases que implementan las operaciones con strings, así como las descripciones de los componentes correspondientes de la Librería Estándar de MQL5.

Las clases de operaciones con strings le ayudarán a ahorrar tiempo en el momento de desarrollar sus propias aplicaciones que lleven a cabo operaciones para procesar textos.

Estas clases de la Librería Estándar de MQL5 se encuentran en el directorio de trabajo del terminal Include\Strings.

Class	Descripción
CString	Clase de las operaciones con cadenas de texto

© 2000-2025, MetaQuotes Ltd.

CString

La clase CString facilita el acceso a las variables de tipo string.

Descripción

La clase CString facilita a sus descendientes el acceso a las funciones de cadena del API MQL5.

Declaración

```
class CString: public CObject
```

Título

```
#include <Strings\String.mqh>
```

Jerarquía de herencia

CObject

CString

Métodos de la clase

Métodos de acceso a los datos	
Str	Obtiene la cadena
Len	Obtiene la longitud de la cadena
Copy	Copia la cadena
Métodos de relleno	
Fill	Llena la cadena con el carácter especificado
Assign	Asigna la cadena
Append	Añade la cadena
Insert	Inserta la cadena
Métodos de comparación	
Compare	Compara la cadena
CompareNoCase	Compara las cadenas sin distinguir entre mayúsculas y minúsculas
Métodos de subcadena	
Left	Obtiene el número especificado de caracteres desde la parte izquierda de la cadena
Right	Obtiene el número especificado de caracteres desde la parte derecha de la cadena

© 2000-2025, MetaQuotes Ltd.

Métodos de acceso a los datos	
Mid	Obtiene el número especificado de caracteres de la cadena
Métodos de recorte/eliminación	
Trim	Borra de la cadena las apariciones finales e iniciales del conjunto de caracteres especificado
TrimLeft	Borra de la cadena las apariciones iniciales del conjunto de caracteres especificado
TrimRight	Borra de la cadena las apariciones finales del conjunto de caracteres especificado
Clear	Libera la cadena
Métodos de conversión	
ToUpper	Convierte la cadena a mayúsculas.
ToLower	Convierte la cadena a minúsculas.
Reverse	Invierte la cadena
Métodos de búsqueda	
Find	Busca la primera coincidencia de la subcadena
FindRev	Busca la última coincidencia de la subcadena
Remove	Borra la subcadena de la cadena
Replace	Reemplaza la subcadena

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type

© 2000-2025, MetaQuotes Ltd.

Str

Obtiene la cadena.

```
string  Str() const;
```

Valor devuelto

Copia de la cadena.

© 2000-2025, MetaQuotes Ltd.

Len

Obtiene la longitud de la cadena.

```
uint  Len() const;
```

Valor devuelto

Longitud de la cadena.

© 2000-2025, MetaQuotes Ltd.

Copy

Copia la cadena por referencia.

```
void  Copy(
   string&  copy      // Referencia
   ) const;
```

Parámetros

copy

[in] Referencia de la cadena a copiar.

Copy

Copia la cadena en la instancia de la clase CString.

```
void  Copy(
   CString*  copy      // Descriptor del objeto
   ) const;
```

Parámetros

copy

[in] Descriptor del objeto de la clase CString.

© 2000-2025, MetaQuotes Ltd.

Fill

Llena la cadena con el carácter especificado.

```
bool  Fill(
   short  character      // Carácter
   )
```

Parámetros

character

[in] Carácter para el relleno.

Valor devuelto

true si se ejecuta correctamente, false si la cadena no se puede llenar.

© 2000-2025, MetaQuotes Ltd.

Assign

Asigna la cadena.

```
void  Assign(
   const string  str      // Cadena a asignar
   )
```

Parámetros

str

[in] Cadena a asignar.

Assign

Asigna la cadena a la instancia de la clase CString.

```
void  Assign(
   CString*  str      // Descriptor del objeto
   )
```

Parámetros

str

[in] Descriptor del objeto de la clase CString a asignar.

© 2000-2025, MetaQuotes Ltd.

Append

Añade la cadena.

```
void  Append(
   const string  str      // Cadena a añadir
   )
```

Parámetros

str

[in] Cadena a añadir.

Append

Añade la cadena a la instancia de la clase CString.

```
void  Append(
   CString*  string      // Descriptor del objeto
   )
```

Parámetros

string

[in] Descriptor del objeto de la clase CString a añadir.

© 2000-2025, MetaQuotes Ltd.

Insert

Inserta la cadena en la posición especificada.

```
uint  Insert(
   uint           pos,      // Posición
   const string   str       // Cadena a insertar
   )
```

Parámetros

pos

[in] Posición a insertar.

str

[in] Cadena a insertar.

Valor devuelto

Longitud de la cadena resultante.

Insert

Inserta la cadena en la posición especificada de la instancia de la clase CString.

```
uint  Insert(
   uint       pos,      // Posición
   CString*   str       // Descriptor del objeto
   )
```

Parámetros

pos

[in] Posición a insertar.

str

[in] Descriptor del objeto de la clase CString a insertar.

Valor devuelto

Longitud de la cadena resultante.

© 2000-2025, MetaQuotes Ltd.

Compare

Compara la cadena.

```
int  Compare(
   const string  str      // Cadena a comparar
   ) const;
```

Parámetros

str

[in] Cadena a comparar.

Valor devuelto

Devuelve 0 si las dos cadenas son iguales, -1 si la cadena es menor que la cadena a comparar, y -1 si es mayor que la cadena a comparar.

Compare

Compara la cadena con una cadena de la instancia de la clase CString.

```
int  Compare(
   CString*  str      // Descriptor del objeto
   ) const;
```

Parámetros

str

[in] Descriptor del objeto de la clase CString a comparar.

Valor devuelto

Devuelve 0 si las dos cadenas son iguales, -1 si la cadena es menor que la cadena a comparar, y -1 si es mayor que la cadena a comparar.

© 2000-2025, MetaQuotes Ltd.

CompareNoCase

Compara las cadenas sin distinguir entre mayúsculas y minúsculas

```
int  CompareNoCase(
   const string  str      // Cadena a comparar
   ) const;
```

Parámetros

str

[in] Cadena a comparar.

Valor devuelto

Devuelve 0 si las dos cadenas son iguales, -1 si la cadena es menor que la cadena a comparar, y -1 si es mayor que la cadena a comparar.

CompareNoCase

Compara la cadena (sin distinguir entre mayúsculas y minúsculas) con una cadena de la instancia de la clase CString.

```
int  CompareNoCase(
   CString*  str      // Descriptor del objeto
   ) const;
```

Parámetros

str

[in] Descriptor del objeto de la clase CString a comparar.

Valor devuelto

Devuelve 0 si las dos cadenas son iguales, -1 si la cadena es menor que la cadena a comparar, y -1 si es mayor que la cadena a comparar.

© 2000-2025, MetaQuotes Ltd.

Left

Obtiene el número especificado de caracteres desde la parte izquierda de la cadena.

```
string  Left(
   uint  count      // Número de caracteres
   )
```

Parámetros

count

[in] Número de caracteres.

Valor devuelto

Subcadena resultante.

© 2000-2025, MetaQuotes Ltd.

Right

Obtiene el número especificado de caracteres desde la parte derecha de la cadena.

```
string  Right(
   uint  count      // Número de caracteres
   )
```

Parámetros

count

[in] Número de caracteres.

Valor devuelto

Subcadena resultante.

© 2000-2025, MetaQuotes Ltd.

Mid

Obtiene el número especificado de caracteres de la cadena.

```
string  Mid(
   uint  pos,        // Posición
   uint  count       // Número de caracteres
   )
```

Parámetros

pos

[in] Posición de la cadena.

count

[in] Número de caracteres.

Valor devuelto

Subcadena resultante.

© 2000-2025, MetaQuotes Ltd.

Trim

Borra de la cadena las apariciones finales e iniciales del conjunto de caracteres especificado (y ' ','\t','\r','\n').

```
int  Trim(
   const string  targets      // Conjunto de caracteres a borrar
   )
```

Parámetros

targets

[in] Conjunto de caracteres a borrar.

Valor devuelto

Número de caracteres borrados.

Ejemplo:

```
//--- ejemplo de CString::Trim
#include <Strings\String.mqh>
//---
void OnStart()
  {
   CString str;
   //---
   str.Assign("   \t\tABCD\r\n");
   printf("Cadena fuente '%s'",str.Str());
   //---
   str.Trim("DA-DA-DA");
   printf("Cadena resultado '%s'",str.Str());
  }
```

© 2000-2025, MetaQuotes Ltd.

TrimLeft

Borra de la cadena las apariciones iniciales del conjunto de caracteres especificado (y ' ','\t','\r','\n').

```
int  TrimLeft(
   const string  targets      // Conjunto de caracteres a borrar
   )
```

Parámetros

targets

[in] Conjunto de caracteres a borrar.

Valor devuelto

Número de caracteres borrados.

© 2000-2025, MetaQuotes Ltd.

TrimRight

Borra de la cadena las apariciones finales del conjunto de caracteres especificado (y ' ','\t','\r','\n').

```
int  TrimRight(
   const string  targets      // Conjunto de caracteres a borrar
   )
```

Parámetros

targets

[in] Conjunto de caracteres a borrar.

Valor devuelto

Número de caracteres borrados.

© 2000-2025, MetaQuotes Ltd.

Clear

Libera la cadena.

```
bool  Clear()
```

Valor devuelto

true si se ejecuta correctamente, false si la cadena no se libera.

© 2000-2025, MetaQuotes Ltd.

ToUpper

Convierte la cadena a mayúsculas.

```
bool  ToUpper()
```

Valor devuelto

true si se ejecuta correctamente; false si la cadena no se puede convertir.

© 2000-2025, MetaQuotes Ltd.

ToLower

Convierte la cadena a minúsculas.

```
bool  ToLower()
```

Valor devuelto

true si se ejecuta correctamente; false si la cadena no se puede convertir.

© 2000-2025, MetaQuotes Ltd.

Reverse

Invierte la cadena.

```
void  Reverse()
```

© 2000-2025, MetaQuotes Ltd.

Find

Busca la primera coincidencia de la subcadena.

```
int  Find(
   uint           start,          // Posición
   const string   substring       // Subcadena a buscar
   ) const;
```

Parámetros

start

[in] El índice del carácter de la cadena por donde comenzar la búsqueda, o 0 para empezar desde el principio.

substring

[in] Subcadena a buscar.

Valor devuelto

El índice del primer carácter que coincide con la subcadena solicitada; -1 si la subcadena no se encuentra.

© 2000-2025, MetaQuotes Ltd.

FindRev

Busca la última coincidencia de la subcadena.

```
int  FindRev(
   const string  substring      // Subcadena
   ) const;
```

Parámetros

substring

[in] Subcadena a buscar.

Valor devuelto

El índice del último carácter que coincide con la subcadena solicitada; -1 si la subcadena no se encuentra.

© 2000-2025, MetaQuotes Ltd.

Remove

Borra la subcadena de la cadena.

```
uint  Remove(
   const string  substring      // Subcadena a borrar
   )
```

Parámetros

substring

[in] Subcadena a buscar.

Valor devuelto

Número de subcadenas borradas.

© 2000-2025, MetaQuotes Ltd.

Replace

Reemplaza la subcadena de la cadena por otra nueva.

```
uint  Replace(
   const string  substring,     // Subcadena a reemplazar
   const string  newstring      // Nueva subcadena
   )
```

Parámetros

substring

[in] Subcadena a buscar.

newstring

[in] Nueva subcadena.

Valor devuelto

Número de subcadenas reemplazadas.

© 2000-2025, MetaQuotes Ltd.

Objetos gráficos

Esta sección contiene detalles técnicos del funcionamiento de las clases de objetos gráficos, así como descripciones de componentes relevantes de la Librería Estándar del lenguaje MQL5.

Las clases de objetos gráficos le ayudarán a ahorrar tiempo en el momento de crear sus propios programas personalizados (tanto scripts como asesores expertos).

Los objetos gráficos de la Librería Estándar MQL5 se encuentran en el directorio de trabajo del terminal, en la carpeta Include\ChartObjects.

Clase/Grupo	Descripción
CChartObject	Clase base de un objeto gráfico
Líneas	Herramientas del grupo "Lines"
Canales	Herramientas del grupo "Channels"
Herramientas de Gann	Clases del grupo "Gann"
Herramientas de Fibonacci	Clases del grupo "Fibonacci"
Herramientas de Elliott	Clases del grupo "Elliott"
Formas	Clases del grupo "Shapes"
Flechas	Clases del grupo "Arrows"
Controles	Clases del grupo "Controls"

© 2000-2025, MetaQuotes Ltd.

CChartObject

La clase CChartObject es la clase base de los objetos de tipo gráfico de la Librería Estándar de MQL5.

Descripción

La clase CChartObject facilita a sus descendientes el acceso a las funciones del API MQL5.

Declaración

```
class CChartObject : public CObject
```

Título

```
#include <ChartObjects\ChartObject.mqh>
```

Jerarquía de herencia

CObject

CChartObject

Descendientes directos

CChartObjectArrow, CChartObjectBitmap, CChartObjectBmpLabel, CChartObjectCycles, CChartObjectElliottWave3, CChartObjectEllipse, CChartObjectFiboArc, CChartObjectFiboFan, CChartObjectFiboTimes, CChartObjectHLine, CChartObjectRectangle, CChartObjectSubChart, CChartObjectText, CChartObjectTrend, CChartObjectTriangle, CChartObjectVLine

Métodos de la clase

Atributos	
ChartId	Obtiene el identificador del gráfico
Window	Obtiene el número de ventanas donde el gráfico es un gráfico
Name	Obtiene/establece el nombre del objeto gráfico
NumPoints	Obtiene el número de puntos de anclaje
Asignación	
Attach	Enlaza el gráfico
SetPoint	Establece el punto de anclaje
Borrado	
Delete	Borra el gráfico
Detach	Libera el gráfico
Desplazamiento	
ShiftObject	Movimiento relativo del objeto
ShiftPoint	Movimiento relativo del punto

© 2000-2025, MetaQuotes Ltd.

Atributos	
Propiedades del objeto	
Time	Obtiene/establece las coordenadas temporales del punto
Price	Obtiene/establece las coordenadas de precio del punto
Color	Obtiene/establece el color del objeto
Style	Obtiene/establece el estilo de línea del objeto
Width	Obtiene/establece la anchura de la línea
BackGround	Obtiene/establece la bandera de dibujo del fondo
Selected	Obtiene/establece la bandera del objeto
Selectable	Obtiene/establece la bandera del objeto seleccionable
Descripción	Obtiene/establece el texto del objeto
Tooltip	Obtiene/establece el tooltip del objeto
Timeframes	Obtiene/establece la máscara de visibilidad de banderas del objeto
Z_Order	Obtiene/establece la prioridad al hacer clic en el gráfico
CreateTime	Obtiene la hora de creación del objeto
Propiedades de los niveles del objeto	
LevelsCount	Obtiene/establece el número de niveles del objeto
LevelColor	Obtiene/establece el color del nivel de la línea
LevelStyle	Obtiene/establece el nivel del estilo de la línea
LevelWidth	Obtiene/establece la anchura del nivel de la línea
LevelValue	Obtiene/establece el nivel
LevelDescription	Obtiene/establece el nivel de texto
Acceso a las funciones del API MQL5	
GetInteger	Obtiene el valor de las propiedades del objeto
SetInteger	Establece las propiedades del objeto
GetDouble	Obtiene el valor de las propiedades del objeto
SetDouble	Establece las propiedades del objeto
GetString	Obtiene el valor de las propiedades del objeto
SetString	Establece las propiedades del objeto

© 2000-2025, MetaQuotes Ltd.

Atributos	
Entrada/Salida	
virtual Save	Método virtual de entrada en el archivo
virtual Load	Método virtual de lectura de un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

© 2000-2025, MetaQuotes Ltd.

ChartId

Obtiene el identificador de la gráfica que tiene un objeto gráfico.

```
long  ChartId() const
```

Valor de retorno

Identificador de la gráfica. Si el objeto no se encuentra, devuelve -1.

Ejemplo:

```
//--- ejemplo de CChartObject::ChartId
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene el identificador de la gráfica del objeto gráfico
   long chart_id=object.ChartId();
  }
```

© 2000-2025, MetaQuotes Ltd.

Window

Obtiene el número de ventanas donde el gráfico es un un objeto.

```
int  Window() const
```

Valor de retorno

Número de ventana donde el objeto gráfico se ubica (0 - ventana principal). Si el objeto no se encuentra, devuelve -1.

Ejemplo:

```
//--- ejemplo de CChartObject::Window
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene la ventana del objeto gráfico
   int window=object.Window();
  }
```

© 2000-2025, MetaQuotes Ltd.

Name (Método Get)

Obtiene el nombre del objeto gráfico.

```
string  Name() const
```

Valor de retorno

Nombre del objeto gráfico enlazado a una instancia de la clase. Si no se encuentra el objeto, devuelve NULL.

Name (Método Set)

Establece el nombre del objeto gráfico.

```
bool  Name(
   string  name      // nombre nuevo
   )
```

Parámetros

name

[in] El nombre nuevo del objeto gráfico.

Valor de retorno

true si se ejecuta correctamente, false - si el nombre no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::Name
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene el nombre del objeto gráfico
   string object_name=object.Name();
   if(object_name!="MiObjetoGrafico")
     {
     //--- establece el nombre del objeto gráfico
     object.Name("MiObjetoGrafico");
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

NumPoints

Obtiene el número de puntos de ancla de un objeto gráfico.

```
int  NumPoints() const
```

Valor de retorno

Número de puntos que enlazan a un objeto gráfico conectado a una instancia de la clase. Si no hay ningún objeto conectado, devuelve 0.

Ejemplo:

```
//--- ejemplo de CChartObject::NumPoints
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtener el número de puntos de un objeto gráfico
   int points=object.NumPoints();
  }
```

© 2000-2025, MetaQuotes Ltd.

Attach

Enlaza un objeto gráfico a una instancia de la clase.

```
bool  Attach(
   long     chart_id,      // Identificador del gráfico
   string   name,          // Nombre del objeto
   int      window,        // Ventana del gráfico
   int      points         // Número de puntos
   )
```

Parámetros

chart_id

[out] Identificador del gráfico.

name

[in] Nombre del objeto gráfico.

window

[in] Número de ventana del gráfico (0 - ventana principal).

points

[in] Número de puntos de ancla del objeto gráfico.

Valor de retorno

true - si se ejecuta correctamente, false - si la operación no se puede ejecutar.

Ejemplo:

```
//--- ejemplo de CChartObject::Attach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- adjuntar objeto gráfico
   if(!object.Attach(ChartID(),"Mi objeto",0,2))
     {
      printf("Error al adjuntar el objeto");
      return;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SetPoint

Establece las nuevas coordenadas del punto de ancla especificado del objeto gráfico.

```
bool  SetPoint(
   int       point,          // Número de punto
   datetime  new_time,       // Coordenada temporal
   double    new_price       // Coordenada de precio
   )
```

Parámetros

point

[in] Número de punto de ancla.

new_time

[in] Valor nuevo de la coordenada temporal del punto de ancla especificado.

new_price

[in] Valor nuevo de la coordenada de precio del punto de ancla especificado.

Valor de retorno

true - si se ejecuta correctamente, false - si no se pueden cambiar las coordenadas del punto.

Ejemplo:

```
//--- ejemplo de CChartObject::SetPoint
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   if(object.NumPoints()>0)
     {
      //--- establece el punto del objeto gráfico
      object.SetPoint(0,CurrTime(),price);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Elimina de la gráfica un objeto gráfico enlazado.

```
bool  Delete()
```

Valor de retorno

true - si se ejecuta correctamente, false - si la operación no se puede ejecutar.

Ejemplo:

```
//--- ejemplo de CChartObject::Delete
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- desvincula el objeto gráfico
   if(!object.Delete())
     {
      printf("Error al borrar el objeto");
      return;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Detach

Desvincula el objeto gráfico.

```
void  Detach()
```

Valor de retorno

Ninguno.

Ejemplo:

```
//--- ejemplo de CChartObject::Detach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- desvincula el objeto gráfico
   object.Detach();
  }
```

© 2000-2025, MetaQuotes Ltd.

ShiftObject

Mueve un objeto gráfico.

```
bool  ShiftObject(
   datetime  d_time,      // Incremento de la coordenada temporal
   double    d_price      // Incremento de la coordenada del precio
   )
```

Parámetros

d_time

[in] Incremento de la coordenada temporal de todos los puntos de ancla.

d_price

[in] Incremento de la coordenada de precio de todos los puntos de ancla.

Valor de retorno

true - si se ejecuta correctamente, false - si el objeto no se puede desplazar.

Ejemplo:

```
//--- ejemplo de CChartObject::ShiftObject
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //--- desplazamiento del objeto gráfico
   object.ShiftObject(d_time,d_price);
  }
```

© 2000-2025, MetaQuotes Ltd.

ShiftPoint

Desplaza el punto de ancla especificado del objeto gráfico.

```
bool  ShiftPoint(
   int       point,       // Número de punto
   datetime  d_time,      // Incremento de la coordenada temporal
   double    d_price      // Incremento de la coordenada del precio
   )
```

Parámetros

point

[in] Número de punto de ancla.

d_time

[in] Incremento de la coordenada temporal de todos los puntos de ancla.

d_price

[in] Incremento de la coordenada de precio de todos los puntos de ancla.

Valor de retorno

true - si se ejecuta correctamente, false - si el punto no se puede desplazar.

Ejemplo:

```
//--- ejemplo de CChartObject::ShiftPoint
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //---
   if(object.NumPoints()>0)
     {
      //--- desplazamiento del punto del objeto gráfico
      object.ShiftPoint(0,d_time,d_price);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Time (Método Get)

Obtiene la coordenada de tiempo del punto de ancla especificado de un objeto gráfico.

```
datetime  Time(
   int  point      // Número de punto
   ) const
```

Parámetros

point

[in] Número de punto de ancla.

Valor de retorno

Coordenada de tiempo del punto de ancla especificado del objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado o el objeto no tiene ese punto, devuelve 0.

Time (Método Set)

Establece la coordenada temporal del punto de ancla especificado de un objeto gráfico.

```
bool  Time(
   int       point,        // Número de punto
   datetime  new_time      // Hora
   )
```

Parámetros

point

[in] Número de punto de ancla.

new_time

[in] Valor nuevo de la coordenada temporal del punto de ancla especificado.

Valor de retorno

true - si se ejecuta correctamente, false - si no se puede cambiar la coordenada temporal.

Ejemplo:

```
//--- ejemplo de CChartObject::Time
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- obtiene la hora correspondiente al punto del objeto gráfico
      datetime point_time=object.Time(i);
      if(point_time==0)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- establece la hora correspondiente al punto del objeto gráfico
          object.Time(i,TimeCurrent());
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

Price (Método Get)

Obtiene la coordenada del precio del punto de ancla especificado de un objeto gráfico.

```
double  Price(
   int  point      // Número de punto
   ) const
```

Parámetros

point

[in] Número de punto de ancla.

Valor de retorno

Coordenada de precio del punto de ancla especificado de un objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado o el objeto no tiene ese punto, devuelve EMPTY_VALUE.

Price (Método Set)

Establece la coordenada del precio del punto de ancla especificado de un objeto gráfico.

```
bool  Price(
   int     point,          // Número de punto
   double  new_price       // Precio
   )
```

Parámetros

point

[in] Número de punto de ancla.

new_price

[in] Valor nuevo de la coordenada del precio del punto de ancla especificado.

Valor de retorno

true - si se ejecuta correctamente, false - si no se puede cambiar la coordenada del precio.

Ejemplo:

```
//--- ejemplo de CChartObject::Price
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- obtiene el precio a partir del punto correspondiente
```

© 2000-2025, MetaQuotes Ltd.

```
      double point_price=object.Price(i);
      if(point_price!=price)
        {
         //--- establece el precio del punto correspondiente
         object.Price(i,price);
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Color (Método Get)

Obtiene el color de la línea del objeto gráfico.

```
color  Color() const
```

Valor de retorno

Color de línea del objeto gráfico enlazado a la instancia de la clase. Si no hay ningún gráfico enlazado, devuelve CLR_NONE.

Color (Método Set)

Establece el color de la línea del objeto gráfico.

```
bool  Color(
   color  new_color      // Color nuevo
   )
```

Parámetros

new_color

[in] Valor nuevo del color de la línea.

Valor de retorno

true - si se ejecuta correctamente, false - si el color de la línea no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::Color
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene el color del objeto gráfico
   color object_color=object.Color();
   if(object_color!=clrRed)
     {
     //--- establece el color del objeto gráfico
     object.Color(clrRed);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Style (Método Get)

Obtiene el estilo de línea del objeto gráfico.

```
ENUM_LINE_STYLE  Style() const
```

Valor de retorno

Estilo de línea del objeto gráfico enlazado a la instancia de la clase. Si no hay ningún objeto enlazado, devuelve WRONG_VALUE.

Style (Método Set)

Establece el estilo de línea del objeto gráfico.

```
bool  Style(
   ENUM_LINE_STYLE  new_style      // Estilo
   )
```

Parámetros

new_style

[in] Nuevo valor del estilo de línea.

Valor de retorno

true - si se ejecuta correctamente, false - si el estilo no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::Style
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtener el estilo del objeto gráfico
   ENUM_LINE_STYLE style=object.Style();
   if(style!=STYLE_SOLID)
     {
      //--- establece el estilo del objeto gráfico
      object.Style(STYLE_SOLID);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Width (Método Get)

Obtiene la anchura de la línea del objeto gráfico.

```
int  Width() const
```

Valor de retorno

La anchura de la línea del objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado, devuelve -1.

Width (Método Set)

Establece la anchura de la línea del objeto gráfico.

```
bool  Width(
   int  new_width      // Grosor
   )
```

Parámetros

new_width

[in] Valor nuevo de la anchura de la línea.

Valor de retorno

true - si se ejecuta correctamente, false - si la anchura no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::Width
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene la anchura del objeto gráfico
   int width=object.Width();
   if(width!=1)
     {
      //--- establece la anchura del objeto gráfico
      object.Width(1);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Background (Método Get)

Obtiene la bandera de dibujar un objeto gráfico en el fondo.

```
bool  Background() const
```

Valor de retorno

Bandera de dibujar en el fondo, un objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado devuelve false.

Background (Método Set)

Establece la bandera de dibujar un objeto gráfico en el fondo.

```
bool  Background(
   bool  background      // Valor de la bandera
   )
```

Parámetros

background

[in] Nuevo valor de la bandera de dibujar un objeto gráfico en el fondo.

Valor de retorno

true - si se ejecuta correctamente, false - si la bandera no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::Background
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene la bandera de fondo del objeto gráfico
   bool background_flag=object.Background();
   if(!background_flag)
     {
     //--- establece la bandera de fondo del objeto gráfico
     object.Background(true);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Selected (Método Get)

Obtiene la bandera que indica si un objeto gráfico está seleccionado. O dicho de otro modo, indica si el objeto gráfico está seleccionado o no.

```
bool  Selected() const
```

Valor de retorno

El estado del objeto, indicando si dicho objeto, enlazado a una instancia de la clase, está seleccionado. Si no hay ningún objeto enlazado, devuelve false.

Selected (Método Set)

Establece la bandera que indica si el objeto gráfico está seleccionado.

```
bool  Selected(
   bool  selected      // Valor de la bandera
   )
```

Parámetros

selected

[in] Nuevo valor de la bandera.

Valor de retorno

true - si se ejecuta correctamente, false - si la bandera no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::Selected
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene la bandera "selected" del objeto gráfico
   bool selected_flag=object.Selected();
   if(selected_flag)
     {
     //--- establece la bandera "selected" del objeto gráfico
     object.Selected(false);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Selectable (Método Get)

Obtiene la bandera que indica si se puede seleccionar un objeto gráfico. O dicho de otro modo, indica si el objeto gráfico se puede seleccionar o no.

```
bool  Selectable() const
```

Valor de retorno

Bandera que indica si se puede seleccionar el objeto enlazado a una instancia de la clase. Si no hay ningún objeto enlazado, devuelve false.

Selectable (Método Set)

Establece la bandera que indica si se puede seleccionar un objeto gráfico.

```
bool  Selectable(
   bool  selectable      // Valor de la bandera
   )
```

Parámetros

selectable

[in] Nuevo valor de la bandera.

Valor de retorno

true - si se ejecuta correctamente, false - si la bandera no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::Selectable
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene la bandera "selectable" del objeto gráfico
   bool selectable_flag=object.Selectable();
   if(selectable_flag)
     {
     //--- establece la bandera "selectable" del objeto gráfico
     object.Selectable(false);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Description (Método Get)

Obtiene una descripción textual de un objeto gráfico.

```
string  Description() const
```

Valor de retorno

Descripción textual del objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado, devuelve NULL.

Description (Método Set)

Establece la descripción textual del objeto gráfico.

```
bool  Description(
   string  text      // Texto
   )
```

Parámetros

text

[in] Nueva descripción textual.

Valor de retorno

true - si se ejecuta correctamente, false - si no se puede cambiar la descripción textual.

Ejemplo:

```
//--- ejemplo de CChartObject::Description
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene la descripción del objeto gráfico
   string description=object.Description();
   if(description=="")
     {
      //--- establece la descripción del objeto gráfico
      object.Description("Mi objeto");
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Tooltip (Método Get)

Obtiene el texto tooltip (descripción emergente) de un objeto gráfico.

```
string  Tooltip() const
```

Valor devuelto

El texto tooltip del objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado, devuelve NULL.

Tooltip (Método Set)

Establece el texto tooltip de un objeto gráfico.

```
bool  Tooltip(
   string  new_tooltip      // nuevo texto de la descripción emergente
   )
```

Parámetros

new_tooltip

[in] Nuevo texto tooltip.

Valor devuelto

true - si se ejecuta correctamente, false si el tooltip no se puede cambiar.

Nota:

Si no se establece la propiedad, entonces se muestra el tooltip generado automáticamente por el terminal. El tooltip se puede desactivar estableciendo el valor "\n" (salto de línea).

© 2000-2025, MetaQuotes Ltd.

Timeframes (Get Method)

Obtiene las banderas de visibilidad de un objeto gráfico.

```
int  Timeframes() const
```

Valor de retorno

Banderas de visibilidad del objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado, devuelve 0.

Timeframes (Método Set)

Establece las banderas de visibilidad de un objeto gráfico.

```
bool  Timeframes(
   int  new_timeframes      // Banderas de visibilidad
   )
```

Parámetros

new_timeframes

[in] Nuevas banderas de visibilidad del objeto gráfico.

Valor de retorno

true - si se ejecuta correctamente, false - si las banderas de visibilidad no se pueden cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::Timeframes
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene los periodos del objeto gráfico
   int timeframes=object.Timeframes();
   if(!(timeframes&OBJ_PERIOD_H1))
     {
      //--- establece los periodos del objeto gráfico
      object.Timeframes(timeframes|OBJ_PERIOD_H1);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Z_Order (Método Get)

Obtiene el valor de la prioridad de clicar sobre un gráfico del objeto gráfico (CHARTEVENT_CLICK).

```
long  Z_Order() const
```

Valor de retorno

Prioridad del objeto gráfico, enlazado a la instancia de la clase. Si no hay ningún objeto enlazado, devuelve 0.

Z_Order (Método Set)

Establece la prioridad de clicar sobre un gráfico del objeto gráfico (CHARTEVENT_CLICK).

```
bool  Z_Order(
   long  value      // nueva prioridad
   )
```

Parámetros

value

[in] Nuevo valor de la prioridad de clicar en un gráfico del objeto gráfico (CHARTEVENT_CLICK).

Valor de retorno

true - si se ejecuta correctamente, false - si la propiedad no se puede cambiar.

Nota

Z_Order es la prioridad de un objeto gráfico para recibir eventos de click en un gráfico (CHARTEVENT_CLICK). La prioridad se puede incrementar estableciendo un valor mayor que 0 (valor predeterminado).

© 2000-2025, MetaQuotes Ltd.

CreateTime

Obtiene la hora de creación de un objeto gráfico.

```
datetime  CreateTime() const
```

Valor de retorno

Hora de creación del objeto gráfico enlazado a la instancia de la clase. Si no hay ningún objeto enlazado, devuelve 0.

Ejemplo:

```
//--- ejemplo de CChartObject::CreateTime
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene la hora de creación del objeto gráfico
   datetime create_time=object.CreateTime();
  }
```

© 2000-2025, MetaQuotes Ltd.

LevelsCount (Método Get)

Obtiene el número de niveles de un objeto gráfico.

```
int  LevelsCount() const
```

Valor de retorno

Número de niveles del objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado, devuelve 0.

LevelsCount (Método Set)

Establece el número de niveles del objeto gráfico.

```
bool  LevelsCount(
   int  levels      // Número de niveles
   )
```

Parámetros

levels

[in] El nuevo número de niveles del objeto gráfico.

Valor de retorno

true - si se ejecuta correctamente, false - si el número de niveles no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::LevelsCount
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene el número de niveles del objeto gráfico
   int levels_count=object.LevelsCount();
   //--- establece el número de niveles del objeto gráfico
   object.LevelsCount(levels_count+1);
  }
```

© 2000-2025, MetaQuotes Ltd.

LevelColor (Método Get)

Obtiene el color de la línea del nivel especificado de un objeto gráfico.

```
color  LevelColor(
   int  level      // Número de nivel
   ) const
```

Parámetros

level

[in] Número de nivel.

Valor de retorno

Color de línea del nivel especificado del objeto gráfico enlazado a la instancia de la clase. Si no hay ningún objeto enlazado o el objeto no tiene el nivel especificado, devuelve CLR_NONE.

LevelColor (Método Set)

Establece el color de la línea del nivel especificado del objeto gráfico.

```
bool  LevelColor(
   int    level,          // Número del nivel
   color  new_color       // Color nuevo
   )
```

Parámetros

level

[in] Número de nivel.

new_color

[in] Nuevo color de la línea del nivel especificado.

Valor de retorno

true - si se ejecuta correctamente, false - si el color de la línea no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::LevelColor
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- obtiene el color del nivel del objeto gráfico
      color level_color=object.LevelColor(i);
      if(level_color!=clrRed)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- establece el color del nivel del objeto gráfico
          object.LevelColor(i,clrRed);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelStyle (Método Get)

Obtiene el estilo de línea del nivel especificado de un objeto gráfico.

```
ENUM_LINE_STYLE  LevelStyle(
   int  level      // Número de nivel
   ) const
```

Parámetros

level

[in] Número de nivel.

Valor de retorno

Estilo de línea del nivel especificado del objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado o el objeto no tiene el nivel especificado, devuelve WRONG_VALUE.

LevelStyle (Método Set)

Establece el estilo de línea del nivel especificado del objeto gráfico.

```
int  LevelStyle(
   int               level,      // Número de nivel
   ENUM_LINE_STYLE   style       // Estilo de línea
   )
```

Parámetros

level

[in] Número de nivel.

style

[in] Nuevo estilo de línea del nivel especificado.

Valor de retorno

true - si se ejecuta correctamente, false - si el estilo no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::LevelStyle
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- obtiene el estilo del nivel del objeto gráfico
      ENUM_LINE_STYLE level_style=object.LevelStyle(i);
      if(level_style!=STYLE_SOLID)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- establece el estilo del nivel del objeto gráfico
          object.LevelStyle(i,STYLE_SOLID);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelWidth (Método Get)

Obtiene la anchura de la línea del nivel especificado de un objeto gráfico.

```
int  LevelWidth(
   int  level      // Número de nivel
   ) const
```

Parámetros

level

[in] Número de nivel.

Valor de retorno

Anchura de la línea del nivel especificado del objeto gráfico enlazado a la instancia de la clase. Si no hay ningún objeto enlazado o el objeto no tiene el nivel especificado, devuelve -1.

LevelWidth (Método Set)

Establece la anchura de la línea del nivel especificado del objeto gráfico.

```
bool  LevelWidth(
   int  level,          // Número de nivel
   int  new_width       // Nueva anchura
   )
```

Parámetros

level

[in] Número de nivel.

new_width

[in] Nueva anchura de línea del nivel especificado.

Valor de retorno

true - si se ejecuta correctamente, false - si la anchura no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::LevelWidth
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- obtiene la anchura del nivel del objeto gráfico
      int level_width=object.LevelWidth(i);
      if(level_width!=1)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- establece la anchura del nivel del objeto gráfico
          object.LevelWidth(i,1);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelValue (Método Get)

Obtiene el valor del nivel especificado de un objeto gráfico.

```
double  LevelValue(
   int  level      // Número de nivel
   ) const
```

Parámetros

level

[in] Número de nivel.

Valor de retorno

El valor del nivel del objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado o el objeto no tiene el nivel especificado, devuelve EMPTY_VALUE.

LevelValue (Método Set)

Establece el valor del nivel especificado del objeto gráfico.

```
bool  LevelValue(
   int     level,          // Número de nivel
   double  new_value       // Valor nuevo
   )
```

Parámetros

level

[in] Número de nivel.

new_value

[in] Nuevo valor del nivel especificado.

Valor de retorno

true - si se ejecuta correctamente, false - si el valor no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::LevelValue
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- obtiene el valor del nivel del objeto gráfico
      double level_value=object.LevelValue(i);
      if(level_value!=0.1*i)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- establece el valor del nivel del objeto gráfico
          object.LevelValue(i,0.1*i);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelDescription (Método Get)

Obtiene una descripción textual del nivel del objeto gráfico.

```
string  LevelDescription(
   int  level       // Número de nivel
   ) const
```

Parámetros

level

[in] Número de niveles del objeto gráfico

Valor de retorno

Descripción textual del nivel del objeto gráfico enlazado a una instancia de la clase. Si no hay ningún objeto enlazado o el objeto no tiene el nivel especificado, devuelve NULL.

LevelDescription (Método Set)

Establece la descripción textual del nivel del objeto gráfico.

```
bool  LevelDescription(
   int     level ,     // Número de nivel
   string  text        // Texto
   )
```

Parámetros

level

[in] Número de nivel del objeto gráfico.

text

[in] Valor nuevo de la descripción (texto) del nivel del objeto gráfico.

Valor de retorno

true - si se ejecuta correctamente, false - si la descripción (texto) no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::LevelDescription
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- obtiene la descripción del nivel del objeto gráfico
      string level_description=object.LevelDescription(i);
      if(level_description=="")
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- establece la descripción del nivel del objeto gráfico
          object.LevelDescription(i,"Level_"+IntegerToString(i));
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

GetInteger

Facilita el acceso a las funciones del API MQL5 ObjectGetInteger() obtiene las propiedades de tipo entero (tipo bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, color) enlazadas a una instancia de la clase gráfica. Hay dos versiones distintas para llamar a esta función:

Obtener el valor de la propiedad sin hacer ninguna comprobación

```
long  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,          // Identificador de las propiedades
   int                           modifier=-1       // Modificador
   ) const
```

Parámetros

prop_id

[in] Identificador de las propiedades de tipo entero del gráfico.

modifier=-1

[in] Modificador (índice) de las propiedades integer.

Valor de retorno

Si se ejecuta correctamente devuelve el valor de la propiedad de tipo entero; en caso de error, devuelve 0.

Obtener el valor de la propiedad llevando a cabo una comprobación

```
bool  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // Identificador de la propiedad intege
   int                           modifier,      // Modificador
   long&                         value          // Enlace a la variable
   ) const
```

Parámetros

prop_id

[in] ID propiedades de tipo integer del objeto.

modifier

[in] Modificador (índice) de la propiedad integer.

value

[out] Referencia a la variable que almacena los valores de las propiedades integer.

Valor de retorno

true - si se ejecuta correctamente, false - si no se puede obtener la propiedad integer.

Ejemplo:

```
//--- ejemplo de CChartObject::GetInteger
#include <ChartObjects\ChartObject.mqh>
//---
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   CChartObject object;
   //--- obtiene el color del objeto gráfico por medio del método sencillo
   printf("El color del objeto es %s",ColorToString(object.GetInteger(OBJPROP_COLOR),t
   //--- obtiene el color del objeto gráfico por medio del método clásico
   long color_value;
   if(!object.GetInteger(OBJPROP_COLOR,0,color_value))
     {
      printf("Error al obtener la propiedad integer %d",GetLastError());
      return;
     }
   else
      printf("El color del objeto es %s",color_value);
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- obtiene la anchura de los niveles por medio del método sencillo
      printf("Nivel %d con anchura %d",i,object.GetInteger(OBJPROP_LEVELWIDTH,i));
      //--- obtiene la anchura de los niveles por medio del método clásico
      long width_value;
      if(!object.GetInteger(OBJPROP_LEVELWIDTH,i,width_value))
        {
         printf("Error al obtener la propiedad integer %d",GetLastError());
         return;
        }
      else
         printf("Nivel %d con anchura %d",i,width_value);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SetInteger

Facilita el acceso a las funciones del API MQL5 ObjectSetInteger() cambia las propiedades integer (tipos bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, color) enlazadas a una instancia de la clase gráfica. Hay dos versiones distintas para llamar a esta función:

Establecer el valor de la propiedad sin utilizar un modificador

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // Identificador de la propiedad intege
   long                           value         // Valor
   )
```

Parámetros

prop_id

[in] ID propiedades de tipo integer del objeto.

value

[in] nuevo valor de las propiedades integer

Establecer el valor de la propiedad indicando el modificador

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // Identificador de la propiedad integ
   int                            modifier,     // Modificador
   long                           value         // Valor
   )
```

Parámetros

prop_id

[in] ID propiedades de tipo integer del objeto.

modifier

[in] Modificador (índice) de la propiedad integer.

value

[in] nuevo valor de las propiedades integer

Valor de retorno

true - si se ejecuta correctamente, false - si no se puede cambiar la propiedad integer.

Ejemplo:

```
//--- ejemplo de CChartObject::SetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- establece el nuevo color del objeto gráfico
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!object.SetInteger(OBJPROP_COLOR,clrRed))
     {
      printf("Error al establecer la propiedad integer %d",GetLastError());
      return;
     }
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- establece la anchura de los niveles
      if(!object.SetInteger(OBJPROP_LEVELWIDTH,i,i))
        {
         printf("Error al establecer la propiedad integer %d",GetLastError());
         return;
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

GetDouble

Facilita el acceso a las funciones del API MQL5 ObjectGetDouble() obtiene los valores de las propiedades double (tipo float y double) del objeto gráfico, asignado a la instancia de la clase. Hay dos versiones distintas para llamar a esta función:

Obtener el valor de la propiedad sin hacer ninguna comprobación

```
double  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,          // Identificador de la propiedad doub
   int                          modifier=-1      // Modificador
   ) const
```

Parámetros

prop_id

[in] Identificador de las propiedades de tipo double.

modifier=-1

[in] Modificador (índice) de las propiedades double.

Valor de retorno

Si se ejecuta correctamente, devuelve el valor de la propiedad de tipo double; en caso de error, devuelve EMPTY_VALUE.

Obtener el valor de la propiedad llevando a cabo una comprobación

```
bool  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,      // Identificador de la propiedad double
   int                          modifier,     // Modificador
   double&                      value         // Enlace a la variable
   ) const
```

Parámetros

prop_id

[in] Identificador de las propiedades de tipo double.

modifier

[in] Modificador (índice) de las propiedades double.

value

[out] Referencia a una variable que almacena las propiedades de tipo double.

Valor de retorno

true - si se ejecuta correctamente, false - si no se pueden obtener las características double.

Ejemplo:

```
//--- ejemplo de CChartObject::GetDouble
#include <ChartObjects\ChartObject.mqh>
//---
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- obtiene los valores de los niveles mediante el método sencillo
      printf("Level %d value=%f",i,object.GetDouble(OBJPROP_LEVELVALUE,i));
      //--- obtiene los valores de los niveles mediante el método clásico
      double value;
      if(!object.SetDouble(OBJPROP_LEVELVALUE,i,value))
        {
         printf("Error al obtener la propiedad double %d",GetLastError());
         return;
        }
      else
         printf("Nivel %d valor=%f",i,value);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SetDouble

Facilita el acceso a las funciones del API MQL5 ObjectSetDouble() cambia las propiedades double (tipo float y tipo double) enlazadas a una instancia de la clase del objeto gráfico. Hay dos versiones distintas para llamar a esta función:

Establecer el valor de la propiedad sin utilizar un modificador

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // Identificador de las propiedades doub
   double                        value         // Valor
   )
```

Parámetros

prop_id

[in] Identificador de las propiedades de tipo double.

value

[in] Valor nuevo de las propiedades double.

Establecer el valor de la propiedad indicando el modificador

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,       // Identificador de la propiedad double
   int                           modifier,      // Modificador
   double                        value          // Valor
   )
```

Parámetros

prop_id

[in] Identificador de las propiedades de tipo double.

modifier

[in] Modificador (índice) de las propiedades double.

value

[in] Nuevo valor de la propiedad double.

Valor de retorno

true - si se ejecuta correctamente, false - si la propiedad double no se puede cambiar.

Ejemplo:

```
//--- ejemplo de CChartObject::SetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- establece el valor del nivel del objeto gráfico
      if(!object.SetDouble(OBJPROP_LEVELVALUE,i,0.1*i))
        {
         printf("Error al establecer la propiedad double %d",GetLastError());
         return;
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

GetString

Facilita el acceso a las funciones del API MQL5 ObjectGetString() obtiene los valores de las propiedades string, se enlaza a una instancia de la clase gráfica. Hay dos versiones distintas para llamar a esta función:

Obtener el valor de la propiedad sin hacer ninguna comprobación

```
string  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,          // Identificador de la propiedad stri
   int                          modifier=-1      // Modificador
   ) const
```

Parámetros

prop_id

[in] Identificador de las propiedades string del objeto gráfico.

modifier=-1

[in] Modificador (índice) de las propiedades string.

Valor de retorno

Valor de la propiedad string.

Obtener el valor de la propiedad llevando a cabo una comprobación

```
bool  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,       // Identificador de la propiedad string
   int                          modifier,      // Modificador
   string&                      value          // Enlace a la variable
   ) const
```

Parámetros

prop_id

[in] Identificador de las propiedades string del objeto gráfico.

modifier

[in] Modificador (índice) de las propiedades string.

value

[out] Referencia a una variable que almacena los valores de las propiedades string.

Valor de retorno

true - si se ejecuta correctamente, false - si no se puede obtener la propiedad string.

Ejemplo:

```
//--- ejemplo de CChartObject::GetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
{
 CChartObject object;
 string       value;
 //--- obtiene el nombre del objeto gráfico por medio del método sencillo
 printf("El nombre del objeto es '%s'",object.GetString(OBJPROP_NAME));
 //--- obtiene el nombre del objeto gráfico por medio del método clásico
 if(!object.GetString(OBJPROP_NAME,0,value))
   {
    printf("Error al obtener la propiedad string %d",GetLastError());
    return;
   }
 else
    printf("El nombre del objeto es '%s'",value);
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- obtiene la descripción de los niveles por medio del método sencillo
    printf("Nivel %d, descripción: '%s'",i,object.GetString(OBJPROP_LEVELTEXT,i));
    //--- obtiene la descripción de los niveles por medio del método clásico
    if(!object.GetString(OBJPROP_LEVELTEXT,i,value))
      {
       printf("Error al obtener la propiedad string %d",GetLastError());
       return;
      }
    else
       printf("Nivel %d, descripción: '%s'",i,value);
   }
}
```

© 2000-2025, MetaQuotes Ltd.

SetString

Facilita el acceso a las funciones del API MQL5 ObjectSetString() cambia las propiedades de tipo string de la instancia de la clase gráfica. Hay dos versiones distintas para llamar a esta función:

Establecer el valor de la propiedad sin utilizar un modificador

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,     // Identificador de la propiedad string
   string                        value        // Valor
   )
```

Parámetros

prop_id

[in] Identificador de las propiedades string del objeto gráfico.

value

[in] Valor nuevo de las propiedades string.

Establecer el valor de la propiedad indicando el modificador

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // Identificador de la propiedad string
   int                           modifier,     // Modificador
   string                        value         // Valor
   )
```

Parámetros

prop_id

[in] Identificador de las propiedades string del objeto gráfico.

modifier

[in] Modificador (índice) de las propiedades string.

value

[in] Valor nuevo de las propiedades string.

Valor de retorno

true - si se ejecuta correctamente, false - si no se puede cambiar la propiedad string.

Ejemplo:

```
//--- ejemplo de CChartObject::SetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- establece el nuevo nombre del objeto gráfico
   if(!object.SetString(OBJPROP_NAME,"MiObjeto"))
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      printf("Error al establecer la propiedad string %d",GetLastError());
      return;
     }
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- establece la descripción de los niveles
      if(!object.SetString(OBJPROP_LEVELTEXT,i,"Nivel_"+IntegerToString(i)))
        {
         printf("Error al establecer la propiedad string %d",GetLastError());
         return;
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en el archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] Manejador del archivo previamente abierto con la función FileOpen (...).

Valor de retorno

true - se se ejecuta correctamente, false - si hay un error.

Ejemplo:

```
//--- ejemplo de CChartObject::Save
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CChartObject object=new CChartObject;
   //--- establece los parámetros del objeto
   //--- . . .
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto con los datos del archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo previamente abierto con la función FileOpen (...).

Valor de retorno

true - se se ejecuta correctamente, false - si hay un error.

Ejemplo:

```
//--- ejemplo de CChartObject::Load
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CChartObject object;
   //--- abrir archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilizar el objeto
   //--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Obtiene el identificador de tipo de un objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador de tipo del objeto (0x8888 para CChartObject).

Ejemplo:

```
//--- ejemplo de CChartObject::Type
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- obtiene el tipo de objeto
   int type=object.Type();
  }
```

© 2000-2025, MetaQuotes Ltd.

Objects Lines

Grupo de objetos gráficos "Líneas".

Esta sección contiene los detalles técnicos para trabajar con el grupo de clases de objetos gráficos llamado "Líneas", así como una descripción de los componentes relevantes de la biblioteca estándar de MQL5.

Nombre de la clase	Objeto
CChartObjectVLine	Objeto gráfico "Línea vertical"
CChartObjectHLine	Objeto gráfico "Línea horizontal"
CChartObjectTrend	Objeto gráfico "Línea de tendencia"
CChartObjectTrendByAngle	Objeto gráfico "Línea de tendencia por ángulo"
CChartObjectCycles	Objeto gráfico "Líneas cíclicas"

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

CChartObjectVLine

La clase CChartObjectVLine facilita el acceso a las propiedades del objeto gráfico "Línea vertical".

Descripción

La clase CChartObjectVLine proporciona acceso a las propiedades del objeto "Línea vertical".

Declaración

```
class CChartObjectVLine : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Jerarquía de herencia

CObject

CChartObject

CChartObjectVLine

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Línea vertical"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Línea vertical".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time           // Coordenada temporal
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time

[in] Coordenada temporal del punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_VLINE para CChartObjectVLine).

© 2000-2025, MetaQuotes Ltd.

CChartObjectHLine

La clase CChartObjectHLine facilita el acceso a las propiedades del objeto gráfico "Línea horizontal".

Descripción

La clase CChartObjectHLine proporciona acceso a las propiedades del objeto "Línea horizontal".

Declaración

```
class CChartObjectHLine : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Jerarquía de herencia

CObject

CChartObject

CChartObjectHLine

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Línea horizontal"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Línea horizontal".

```
bool  Create(
   long    chart_id,      // Identificador del gráfico
   string  name,          // Nombre del objeto
   long    window,        // Ventana del gráfico
   double  price          // Coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

price

[in] Coordenada del precio del punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_HLINE para CChartObjectHLine).

© 2000-2025, MetaQuotes Ltd.

CChartObjectTrend

La clase CChartObjectTrend facilita el acceso a las propiedades del objeto gráfico "Línea de tendencia".

Descripción

La clase CChartObjectTrend proporciona acceso a las propiedades del objeto "Línea de tendencia".

Declaración

```
class CChartObjectTrend : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Jerarquía de herencia

CObject

CChartObject

CChartObjectTrend

Descendientes directos

CChartObjectChannel, CChartObjectFibo, CChartObjectFiboChannel, CChartObjectFiboExpansion, CChartObjectGannFan, CChartObjectGannGrid, CChartObjectPitchfork, CChartObjectRegression, CChartObjectStdDevChannel, CChartObjectTrendByAngle

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Línea de tendencia"
Propiedades	
RayLeft	Obtiene/Establece la propiedad "Rayo a la izquierda"
RayRight	Obtiene/Establece la propiedad "Rayo a la derecha"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Línea de tendencia".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2         // Segunda coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

RayLeft (Método Get)

Obtiene el valor de la propiedad "Rayo a la izquierda".

```
bool  RayLeft() const
```

Valor devuelto

El valor de la propiedad "Rayo a la izquierda", asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

RayLeft (Método Set)

Establece el nuevo valor de la bandera de la propiedad "Rayo a la izquierda".

```
bool  RayLeft(
   bool  ray      // bandera
   )
```

Parámetros

ray

[in] Nuevo valor de la propiedad "Rayo a la izquierda".

Valor devuelto

true si se ejecuta correctamente, false si la bandera no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

RayRight (Método Get)

Obtiene el valor de la propiedad "Rayo a la derecha".

```
bool  RayRight() const
```

Valor devuelto

El valor de la propiedad "Rayo a la derecha", asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

RayRight (Método Set)

Establece el nuevo valor de la bandera de la propiedad "Rayo a la derecha".

```
bool  RayRight(
   bool  ray      // bandera
   )
```

Parámetros

ray

[in] Nuevo valor de la propiedad "Rayo a la derecha".

Valor devuelto

true si se ejecuta correctamente, false si la bandera no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo previamente abierto con la función FileOpen(...) .

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo abierto previamente con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_TREND para CChartObjectTrend).

© 2000-2025, MetaQuotes Ltd.

CChartObjectTrendByAngle

La clase CChartObjectTrendByAngle facilita el acceso a las propiedades del objeto gráfico "Línea de tendencia por ángulo".

Descripción

La clase CChartObjectTrendByAngle proporciona acceso a las propiedades del objeto "Línea de tendencia por ángulo".

Declaración

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Jerarquía de herencia

CObject
CChartObject
CChartObjectTrend
CChartObjectTrendByAngle

Descendientes directos

CChartObjectGannLine

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Línea de tendencia por ángulo"
Propiedades	
Angle	Obtiene/Establece la propiedad "Ángulo"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Línea de tendencia por ángulo"

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   long      window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2         // Segunda coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Angle (Método Get)

Obtiene el valor de la propiedad "Ángulo".

```
double  Angle() const
```

Valor devuelto

El valor de la propiedad "Ángulo", asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

Angle (Método Set)

Establece el nuevo valor de la propiedad "Ángulo".

```
bool  Angle(
   double  angle      // Ángulo
   )
```

Parámetros

angle

[in] Nuevo valor de la propiedad "Ángulo".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_TRENDBYANGLE para CChartObjectTrendByAngle).

© 2000-2025, MetaQuotes Ltd.

CChartObjectCycles

La clase CChartObjectCycles facilita el acceso a las propiedades del objeto gráfico "Líneas cíclicas".

Descripción

La clase CChartObjectCycles proporciona acceso a las propiedades del objeto "Cyclic Lines".

Declaración

```
class CChartObjectCycles : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Jerarquía de herencia

CObject

CChartObject

CChartObjectCycles

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Líneas cíclicas"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Líneas cíclicas"

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   long      window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2         // Segunda coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_CYCLES para CChartObjectCycles).

© 2000-2025, MetaQuotes Ltd.

Objects Channels

Grupo de objetos gráficos "Canales".

Esta sección contiene los detalles técnicos para trabajar con el grupo de clases de objetos gráficos llamado "Canales", así como una descripción de los componentes relevantes de la biblioteca estándar de MQL5.

Nombre de la clase	Objeto
CChartObjectChannel	Objeto gráfico "Canal equidistante"
CChartObjectRegression	Objeto gráfico "Canal de regresión lineal"
CChartObjectStdDevChannel	Objeto gráfico "Canal de desviación estándar"
CChartObjectPitchfork	Objeto gráfico "Horquilla de Andrew"

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

CChartObjectChannel

La clase CChartObjectChannel facilita el acceso a las propiedades del objeto gráfico "Canal Equidistante".

Descripción

La clase CChartObjectChannel proporciona acceso a las propiedades del objeto "Canal Equidistante".

Declaración

```
class CChartObjectChannel : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectChannel

Métodos de la clase

Creación	
Create	Crea el objeto gráfico "Canal Equidistante"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ver también

© 2000-2025, MetaQuotes Ltd.

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Canal Equidistante"

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Coordenada temporal del primer punto de anclaje
   double    price1,        // Coordenada del precio del primer punto de anclaje
   datetime  time2,         // Coordenada temporal del segundo punto de anclaje
   double    price2,        // Coordenada del precio del segundo punto de anclaje
   datetime  time3,         // Coordenada temporal del tercer punto de anclaje
   double    price3         // Coordenada del precio del tercer punto de anclaje
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

time3

[in] Coordenada temporal del tercer punto de anclaje.

price3

[in] Coordenada del precio del tercer punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_CHANNEL para CChartObjectChannel).

© 2000-2025, MetaQuotes Ltd.

CChartObjectRegression

La clase CChartObjectRegression facilita el acceso a las propiedades del objeto gráfico "Canal de regresión lineal".

Descripción

CChartObjectRegression proporciona acceso a las propiedades del objeto "Canal de regresión lineal".

Declaración

```
class CChartObjectRegression : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectRegression

Métodos de la clase

Creación	
Create	Crea el objeto gráfico "Canal de regresión lineal"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ver también

© 2000-2025, MetaQuotes Ltd.

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Canal de regresión lineal"

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   long      window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   datetime  time2          // Segunda coordenada temporal
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_REGRESSION para CChartObjectRegression).

© 2000-2025, MetaQuotes Ltd.

CChartObjectStdDevChannel

La clase CChartObjectStdDevChannel facilita el acceso a las propiedades del objeto gráfico "Canal de desviación estándar".

Descripción

La clase CChartObjectStdDevChannel proporciona acceso a las propiedades del objeto "Canal de desviación estándar".

Declaración

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectStdDevChannel

Métodos de la clase

Creación	
Create	Crea el objeto gráfico "Canal de desviación estándar"
Propiedades	
Deviations	Establece/Obtiene la propiedad "Desviación"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Canal de desviación estándar"

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   datetime  time2,         // Segunda coordenada temporal
   double    deviation      // Desviación
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

deviation

[in] Valor numérico de la propiedad "Desviación".

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Deviation (Método Get)

Obtiene el valor numérico de la propiedad "Desviación".

```
double  Deviation() const
```

Valor devuelto

Valor numérico de la propiedad "Desviación", asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

Deviation (Método Set)

Establece el valor numérico de la propiedad "Desviación".

```
bool  Deviation(
   double  deviation      // Desviación
   )
```

Parámetros

deviation

[in] Valor nuevo de la propiedad "Desviación".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo previamente abierto con la función FileOpen(...) .

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo previamente abierto con la función FileOpen(...) .

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_STDDEVCHANNEL para CChartObjectStdDevChannel).

© 2000-2025, MetaQuotes Ltd.

CChartObjectPitchfork

CChartObjectPitchfork facilita el acceso a las propiedades del objeto gráfico "Horquilla de Andrew".

Descripción

CChartObjectPitchfork proporciona acceso a las propiedades del objeto "Horquilla de Andrew".

Declaración

```
class CChartObjectPitchfork : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectPitchfork

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Horquilla de Andrew"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Horquilla de Andrew"

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   long      window,        // Ventana del gráfico
   datetime  time1,         // Coordenada temporal del primer punto de anclaje
   double    price1,        // Coordenada del precio del primer punto de anclaje
   datetime  time2,         // Coordenada temporal del segundo punto de anclaje
   double    price2,        // Coordenada del precio del segundo punto de anclaje
   datetime  time3,         // Coordenada temporal del tercer punto de anclaje
   double    price3         // Coordenada del precio del tercer punto de anclaje
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del primer punto de anclaje.

time3

[in] Coordenada temporal del tercer punto de anclaje.

price3

[in] Coordenada del precio del tercer punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_PITCHFORK para CChartObjectPitchfork).

© 2000-2025, MetaQuotes Ltd.

Gann Tools

Grupo de objetos gráficos "Herramientas de Gann".

Esta sección contiene los detalles técnicos para trabajar con el grupo de clases de objetos gráficos llamado "Herramientas de Gann", así como una descripción de los componentes relevantes de la biblioteca estándar de MQL5.

Nombre de la clase	Objeto
CChartObjectGannLine	Objeto gráfico "Línea de Gann"
CChartObjectGannFan	Objeto gráfico "Abanico de Gann"
CChartObjectGannGrid	Objeto gráfico "Rejilla de Gann"

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

CChartObjectGannLine

La clase CChartObjectGannLine facilita el acceso a las propiedades del objeto gráfico "Línea de Gann".

Descripción

La clase CChartObjectGannLine proporciona acceso a las propiedades del objeto "Línea de Gann".

Declaración

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

Título

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectTrendByAngle
        CChartObjectGannLine

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Línea de Gann"
Propiedades	
PipsPerBar	Obtiene/Establece la propiedad "Escala"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Métodos heredados de la clase CChartObjectTrendByAngle

Angle, Angle, Create

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Línea de Gann".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    ppb            // Pips por barra
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

ppb

[in] Pips por barra.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

PipsPerBar (Método Get)

Obtiene el valor de la propiedad "Pips por barra".

```
double  PipsPerBar() const
```

Valor devuelto

Valor de la propiedad "Pips por barra" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

PipsPerBar (Método Set)

Establece el nuevo valor de la propiedad "Pips por barra".

```
bool  PipsPerBar(
   double  ppb       // Pips por barra
   )
```

Parámetros

ppb

[in] Nuevo valor de la propiedad "Pips por barra".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_GANNLINE para CChartObjectGannLine).

© 2000-2025, MetaQuotes Ltd.

CChartObjectGannFan

La clase CChartObjectGannFan facilita el acceso a las propiedades del objeto gráfico "Abanico de Gann".

Descripción

La clase CChartObjectGannFan proporciona acceso a las propiedades del objeto "Abanico de Gann".

Declaración

```
class CChartObjectGannFan : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannFan

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Abanico de Gann"
Propiedades	
PipsPerBar	Obtiene/Establece la propiedad "Pips por barra"
Downtrend	Obtiene/Establece la propiedad "Tendencia bajista"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Abanico de Gann".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    ppb            // Pips por barra
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

ppb

[in] Pips por barra.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

PipsPerBar (Método Get)

Obtiene el valor de la propiedad "Pips por barra".

```
double  PipsPerBar() const
```

Valor devuelto

Valor de la propiedad "Pips por barra" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

PipsPerBar (Método Set)

Establece el nuevo valor de la propiedad "Pips por barra".

```
bool  PipsPerBar(
   double  ppb       // Pips por barra
   )
```

Parámetros

ppb

[in] Nuevo valor de la propiedad "Pips por barra".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Downtrend (Método Get)

Obtiene el valor de la propiedad "Tendencia bajista".

```
bool  Downtrend() const
```

Valor devuelto

Valor de la propiedad "Tendencia bajista" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

Downtrend (Método Set)

Establece el nuevo valor de la propiedad "Tendencia bajista".

```
bool  Downtrend(
   bool  downtrend      // Valor de la bandera
   )
```

Parámetros

downtrend

[in] Nuevo valor de la propiedad "Tendencia bajista".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_GANNFAN para CChartObjectGannFan).

© 2000-2025, MetaQuotes Ltd.

CChartObjectGannGrid

La clase CChartObjectGannGrid facilita el acceso a las propiedades del objeto gráfico "Rejilla de Gann".

Descripción

La clase CChartObjectGannGrid proporciona acceso a las propiedades del objeto "Rejilla de Gann".

Declaración

```
class CChartObjectGannGrid : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannGrid

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Rejilla de Gann"
Propiedades	
PipsPerBar	Obtiene/Establece la propiedad "Pips por barra"
Downtrend	Obtiene/Establece la propiedad "Tendencia bajista"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Rejilla de Gann".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    ppb            // Pips por barra
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

ppb

[in] Pips por barra.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

PipsPerBar (Método Get)

Obtiene el valor de la propiedad "Pips por barra".

```
double  PipsPerBar() const
```

Valor devuelto

Valor de la propiedad "Pips por barra" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

PipsPerBar (Método Set)

Establece el nuevo valor de la propiedad "Pips por barra".

```
bool  PipsPerBar(
   double  ppb       // Pips por barra
   )
```

Parámetros

ppb

[in] Nuevo valor de la propiedad "Pips por barra".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Downtrend (Método Get)

Obtiene el valor de la propiedad "Tendencia bajista".

```
bool  Downtrend() const
```

Valor devuelto

Valor de la propiedad "Tendencia bajista" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

Downtrend (Método Set)

Establece el nuevo valor de la propiedad "Tendencia bajista".

```
bool  Downtrend(
   bool  downtrend      // Valor de la bandera
   )
```

Parámetros

downtrend

[in] Nuevo valor de la propiedad "Tendencia bajista".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_GANNGRID para CChartObjectGannGrid).

© 2000-2025, MetaQuotes Ltd.

Fibonacci Tools

Grupo de objetos gráficos "Herramientas de Fibonacci".

Esta sección contiene los detalles técnicos para trabajar con el grupo de clases de objetos gráficos llamado "Herramientas de Fibonacci", así como una descripción de los componentes relevantes de la biblioteca estándar de MQL5.

Nombre de la clase	Objeto
CChartObjectFibo	Objeto gráfico "Retroceso de Fibonacci"
CChartObjectFiboTimes	Objeto gráfico "Zonas horarias de Fibonacci"
CChartObjectFiboFan	Objeto gráfico "Abanico de Fibonacci"
CChartObjectFiboArc	Objeto gráfico "Arco de Fibonacci"
CChartObjectFiboChannel	Objeto gráfico "Canal de Fibonacci"
CChartObjectFiboExpansion	Objeto gráfico "Expansión de Fibonacci"

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

CChartObjectFibo

La clase CChartObjectFibo facilita el acceso a las propiedades del objeto gráfico "Retroceso de Fibonacci".

Descripción

La clase CChartObjectFibo proporciona acceso a las propiedades del objeto "Retroceso de Fibonacci".

Declaración

```
class CChartObjectFibo : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFibo

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Retroceso de Fibonacci"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ver también

© 2000-2025, MetaQuotes Ltd.

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Retroceso de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2         // Segunda coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_FIBO para CChartObjectFibo).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboTimes

La clase CChartObjectFiboTimes facilita el acceso a las propiedades del objeto gráfico "Zonas horarias de Fibonacci".

Descripción

La clase CChartObjectFiboTimes proporciona acceso a las propiedades del objeto "Zonas horarias de Fibonacci".

Declaración

```
class CChartObjectFiboTimes : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Jerarquía de herencia

CObject

CChartObject

CChartObjectFiboTimes

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Zonas horarias de Fibonacci"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Zonas horarias de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2         // Segunda coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_FIBOTIMES para CChartObjectFiboTimes).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboFan

La clase CChartObjectFiboFan facilita el acceso a las propiedades del objeto gráfico "Abanico de Fibonacci".

Descripción

La clase CChartObjectFiboFan proporciona acceso a las propiedades del objeto "Abanico de Fibonacci".

Declaración

```
class CChartObjectFiboFan : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectFiboFan

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Abanico de Fibonacci"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Abanico de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2         // Segunda coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_FIBOFAN para CChartObjectFiboFan).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboArc

La clase CChartObjectFiboArc facilita el acceso a las propiedades del objeto gráfico "Arco de Fibonacci".

Descripción

La clase CChartObjectFiboArc proporciona acceso a las propiedades del objeto "Arco de Fibonacci".

Declaración

```
class CChartObjectFiboArc : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectFiboArc

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Arco de Fibonacci"
Propiedades	
Scale	Obtiene/Establece la propiedad "Escala"
Ellipse	Obtiene/Establece la propiedad "Elipse"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Arco de Fibonacci"

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2,        // Segunda coordenada del precio
   double    scale          // Escala
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

scale

[in] Escala.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Scale (Método Get)

Obtiene el valor de la propiedad "Escala".

```
double  Scale() const
```

Valor devuelto

Valor de la propiedad "Escala" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

Scale (Método Set)

Establece el nuevo valor de la propiedad "Escala".

```
bool  Scale(
   double  scale      // Escala
   )
```

Parámetros

scale

[in] Nuevo valor de la propiedad "Escala".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Ellipse (Método Get)

Obtiene el valor de la propiedad "Elipse".

```
bool  Ellipse() const
```

Valor devuelto

Valor de la propiedad "Elipse" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

Ellipse (Método Set)

Establece el nuevo valor de la bandera de la propiedad "Elipse".

```
bool  Ellipse(
   bool  ellipse      // valor de la bandera
   )
```

Parámetros

ellipse

[in] Nuevo valor de la propiedad "Escala".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_FIBOARC para CChartObjectFiboArc).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboChannel

La clase CChartObjectFiboChannel facilita el acceso a las propiedades del objeto gráfico "Canal de Fibonacci".

Descripción

La clase CChartObjectFiboChannel proporciona acceso a las propiedades del objeto "Canal de Fibonacci".

Declaración

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboChannel

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Canal de Fibonacci"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ver también

© 2000-2025, MetaQuotes Ltd.

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Canal de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2,        // Segunda coordenada del precio
   datetime  time3,         // Tercera coordenada temporal
   double    price3         // Tercera coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

time3

[in] Coordenada temporal del tercer punto de anclaje.

price3

[in] Coordenada del precio del tercer punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_FIBOCHANNEL para CChartObjectFiboChannel).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboExpansion

La clase CChartObjectFiboExpansion facilita el acceso a las propiedades del objeto gráfico "Expansión de Fibonacci".

Descripción

La clase CChartObjectFiboExpansion proporciona acceso a las propiedades del objeto "Expansión de Fibonacci".

Declaración

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboExpansion

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Expansión de Fibonacci"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ver también

© 2000-2025, MetaQuotes Ltd.

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Expansión de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2,        // Segunda coordenada del precio
   datetime  time3,         // Tercera coordenada temporal
   double    price3         // Tercera coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

time3

[in] Coordenada temporal del tercer punto de anclaje.

price3

[in] Coordenada del precio del tercer punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_EXPANSION para CChartObjectFiboExpansion).

© 2000-2025, MetaQuotes Ltd.

Elliott Tools

Grupo de objetos gráficos "Herramientas de Elliott".

Esta sección contiene los detalles técnicos para trabajar con el grupo de clases de objetos gráficos llamado "Herramientas de Elliott", así como una descripción de los componentes relevantes de la biblioteca estándar de MQL5.

Nombre de la clase	Objeto
CChartObjectElliottWave3	Objeto gráfico "Onda correctiva"
CChartObjectElliottWave5	Objeto gráfico "Onda de impulso"

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

CChartObjectElliottWave3

La clase CChartObjectElliottWave3 facilita el acceso a las propiedades del objeto gráfico "Onda correctiva".

Descripción

La clase CChartObjectElliottWave3 proporciona acceso a las propiedades del objeto "Onda correctiva".

Declaración

```
class CChartObjectElliottWave3 : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Jerarquía de herencia

CObject

CChartObject

CChartObjectElliottWave3

Descendientes directos

CChartObjectElliottWave5

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Onda correctiva"
Propiedades	
Degree	Obtiene/Establece la propiedad "Grado"
Lines	Obtiene/Establece la propiedad "Líneas"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Onda correctiva".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2,        // Segunda coordenada del precio
   datetime  time3,         // Tercera coordenada temporal
   double    price3         // Tercera coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

time3

[in] Coordenada temporal del tercer punto de anclaje.

price3

[in] Coordenada temporal del tercer punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Degree (Método Get)

Obtiene el valor de la propiedad "Grado".

```
ENUM_ELLIOT_WAVE_DEGREE  Degree() const
```

Valor devuelto

Valor de la propiedad "Grado" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve WRONG_VALUE.

Degree (Método Set)

Establece el valor nuevo de la propiedad "Grado".

```
bool  Degree(
   ENUM_ELLIOT_WAVE_DEGREE  degree      // valor de la propiedad
   )
```

Parámetros

degree

[in] Nuevo valor de la propiedad "Grado".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Lines (Método Get)

Obtiene el valor de la propiedad "Líneas".

```
bool  Lines() const
```

Valor devuelto

Valor de la propiedad "Líneas" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

Lines (Método Set)

Establece el nuevo valor de la propiedad "Líneas".

```
bool  Lines(
   bool  lines      // valor de la bandera
   )
```

Parámetros

lines

[in] Nuevo valor de la propiedad "Líneas".

Valor devuelto

true si se ejecuta correctamente, false si la bandera no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_ELLIOTWAVE3 para CChartObjectElliottWave3).

© 2000-2025, MetaQuotes Ltd.

CChartObjectElliottWave5

La clase CChartObjectElliottWave5 facilita el acceso a las propiedades del objeto gráfico "Onda de impulso".

Descripción

La clase CChartObjectElliottWave5 proporciona acceso a las propiedades del objeto "Onda de impulso".

Declaración

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

Título

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectElliottWave3
      CChartObjectElliottWave5

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Onda de impulso"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectElliottWave3

Degree, Degree, Lines, Lines, Create, Save, Load

Ver también

© 2000-2025, MetaQuotes Ltd.

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Onda de impulso".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2,        // Segunda coordenada del precio
   datetime  time3,         // Tercera coordenada temporal
   double    price3,        // Tercera coordenada del precio
   datetime  time4,         // Cuarta coordenada temporal
   double    price4,        // Cuarta coordenada del precio
   datetime  time5          // Quinta coordenada temporal
   double    price5,        // Quinta coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

time3

[in] Coordenada temporal del tercer punto de anclaje.

price3

[in] Coordenada del precio del tercer punto de anclaje.

time4

[in] Coordenada temporal del cuarto punto de anclaje.

© 2000-2025, MetaQuotes Ltd.

price4

[in] Coordenada del precio del cuarto punto de anclaje.

time5

[in] Coordenada temporal del quinto punto de anclaje.

price5

[in] Coordenada del precio del quinto punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_ELLIOTWAVE5 para CChartObjectElliottWave5).

© 2000-2025, MetaQuotes Ltd.

Objects Shapes

Grupo de objetos gráficos "Formas".

Esta sección contiene los detalles técnicos para trabajar con el grupo de clases de objetos gráficos llamado "Formas", así como una descripción de los componentes relevantes de la biblioteca estándar de MQL5.

Nombre de la clase	Objeto
CChartObjectRectangle	Objeto gráfico "Rectángulo"
CChartObjectTriangle	Objeto gráfico "Triángulo"
CChartObjectEllipse	Objeto gráfico "Elipse"

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

CChartObjectRectangle

La clase CChartObjectRectangle facilita el acceso a las propiedades del objeto gráfico "Rectángulo".

Descripción

La clase CChartObjectRectangle proporciona acceso a las propiedades del objeto "Rectángulo".

Declaración

```
class CChartObjectRectangle : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Jerarquía de herencia

CObject

CChartObject

CChartObjectRectangle

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Rectángulo"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Rectángulo".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   long      window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2         // Segunda coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, 0 en caso de error

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_RECTANGLE para CChartObjectRectangle).

© 2000-2025, MetaQuotes Ltd.

CChartObjectTriangle

La clase CChartObjectTriangle facilita el acceso a las propiedades del objeto gráfico "Triángulo".

Descripción

La clase CChartObjectTriangle proporciona acceso a las propiedades del objeto "Triángulo".

Declaración

```
class CChartObjectTriangle : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Jerarquía de herencia

CObject

CChartObject

CChartObjectTriangle

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Triángulo"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Triángulo".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   long      window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2,        // Segunda coordenada del precio
   datetime  time3,         // Tercera coordenada temporal
   double    price3         // Tercera coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

time3

[in] Coordenada temporal del tercer punto de anclaje.

price3

[in] Coordenada del precio del tercer punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_TRIANGLE para CChartObjectTriangle).

© 2000-2025, MetaQuotes Ltd.

CChartObjectEllipse

La clase CChartObjectEllipse facilita el acceso a las propiedades del objeto gráfico "Elipse".

Descripción

La clase CChartObjectEllipse proporciona acceso a las propiedades del objeto "Elipse".

Declaración

```
class CChartObjectEllipse : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Jerarquía de herencia

CObject

CChartObject

CChartObjectEllipse

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Elipse"
Entrada/salida	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Elipse".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time1,         // Primera coordenada temporal
   double    price1,        // Primera coordenada del precio
   datetime  time2,         // Segunda coordenada temporal
   double    price2,        // Segunda coordenada del precio
   datetime  time3,         // Tercera coordenada temporal
   double    price3         // Tercera coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time1

[in] Coordenada temporal del primer punto de anclaje.

price1

[in] Coordenada del precio del primer punto de anclaje.

time2

[in] Coordenada temporal del segundo punto de anclaje.

price2

[in] Coordenada del precio del segundo punto de anclaje.

time3

[in] Coordenada temporal del tercer punto de anclaje.

price3

[in] Coordenada del precio del tercer punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto del objeto gráfico.

```
int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_ELLIPSE para CChartObjectEllipse).

© 2000-2025, MetaQuotes Ltd.

Objects Arrows

Grupo de objetos gráficos "Flechas".

Esta sección contiene los detalles técnicos para trabajar con el grupo de clases de objetos gráficos llamado "Flechas", así como una descripción de los componentes relevantes de la biblioteca estándar de MQL5. En esencia, la flecha es un icono que se muestra al usuario y que se corresponde con un código determinado. Hay dos tipos de objetos gráficos "Flecha" que muestran iconos en los gráficos:

- El objeto "Flecha", que permite especificar el código del icono mostrado.
- Objetos de grupo que muestran ciertos tipos de iconos (y el correspondiente código fijo).

La clase de la flecha muestra iconos de código arbitrario.

Nombre de la clase	Nombre del objeto flecha
CChartObjectArrow	Arrow

Clases para el código fijo del icono de la flecha.

Nombre de la clase	Nombre del objeto flecha
CChartObjectArrowCheck	Comprobar
CChartObjectArrowDown	Flecha hacia arriba
CChartObjectArrowUp	Flecha hacia abajo
CChartObjectArrowStop	Símbolo Stop
CChartObjectArrowThumbDown	Pulgar hacia arriba
CChartObjectArrowThumbUp	Pulgar hacia abajo
CChartObjectArrowLeftPrice	Etiqueta del precio izquierda
CChartObjectArrowRightPrice	Etiqueta del precio derecha

Ver también

Tipos de objetos, Modos de enlace de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

CChartObjectArrow

La clase CChartObjectArrow facilita el acceso a las propiedades del objeto gráfico "Flecha".

Descripción

La clase CChartObjectArrow proporciona acceso a los descendientes de objetos "Flecha" a las propiedades comunes.

Declaración

```
class CChartObjectArrow : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectArrow

Descendientes directos

CChartObjectArrowCheck, CChartObjectArrowDown, CChartObjectArrowLeftPrice, CChartObjectArrowRightPrice, CChartObjectArrowStop, CChartObjectArrowThumbDown, CChartObjectArrowThumbUp, CChartObjectArrowUp

Métodos de la clase

Creación	
Create	Crea el objeto gráfico "Flecha"
Propiedades	
ArrowCode	Obtiene/Establece la propiedad "Código de la flecha"
Anchor	Obtiene/Establece la propiedad "Anclaje"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Ver también

Tipos de objetos, Modos de enlace de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea un objeto gráfico "Flecha".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time,          // Hora
   double    price,         // Precio
   char      code           // Código de la flecha
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Nombre del objeto (Tiene que ser único).

window

[in] Número de la ventana del gráfico (0 - ventana base).

time

[in] Coordenada temporal.

price

[in] Coordenada de precio.

code

[in] Código de la "Flecha" (Wingdings).

Valor devuelto

true - si se ejecuta correctamente, false en caso contrario.

Ejemplo:

```
//--- ejemplo de CChartObjectArrow::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
//--- establece los parámetros del objeto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Flecha",0,TimeCurrent(),price,181))
     {
      //--- error en la creación de la flecha
      printf("Error en la creación de la flecha: %d!",GetLastError());
      //---
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- utiliza la flecha
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrowCode (Método Get)

Obtiene el código del símbolo del objeto "Flecha".

```
char  ArrowCode() const
```

Valor devuelto

Código del símbolo del objeto "Flecha", asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

ArrowCode (Método Set)

Establece el código del símbolo de "Flecha"

```
bool  ArrowCode(
   char  code      // Valor del código
   )
```

Parámetros

code

[in] nuevo valor del código "flecha" (Wingdings).

Valor devuelto

true - si se ejecuta correctamente, false - si el código no ha cambiado.

Ejemplo:

```
//--- ejemplo de CChartObjectArrow::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   char              code=181;
//--- establece los parámetros del objeto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Flecha",0,TimeCurrent(),price,code))
     {
      //--- error en la creación de la flecha
      printf("Error en la creación de la flecha: %d!",GetLastError());
      //---
      return;
     }
//--- utiliza la flecha
//--- . . .
//--- obtiene el código de la flecha
   if(arrow.ArrowCode()!=code)
     {
     //--- establece el código de la flecha
```

© 2000-2025, MetaQuotes Ltd.

```
      arrow.ArrowCode(code);
      }
//--- utiliza la flecha
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Anchor (Método Get)

Obtiene el tipo de anclaje del objeto "Flecha"

```
ENUM_ARROW_ANCHOR  Anchor() const
```

Valor devuelto

Tipo de anclaje del objeto "Flecha", asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve WRONG_VALUE.

Anchor (Método Set)

Establece el tipo de anclaje del objeto "Arrow"

```
bool  Anchor(
   ENUM_ARROW_ANCHOR  anchor      // nuevo tipo de anclaje
   )
```

Parámetros

anchor

[in] Nuevo tipo de anclaje

Valor devuelto

true si se ejecuta correctamente, false si el tipo de anclaje no ha cambiado

Ejemplo:

```
//--- ejemplo de CChartObject::Anchor
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM;
//--- establece los parámetros del objeto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Flecha",0,TimeCurrent(),price,181))
     {
      //--- error en la creación de la flecha
      printf("Error en la creación de la flecha: %d!",GetLastError());
      //---
      return;
     }
//--- obtiene el anclaje de la flecha
   if(arrow.Anchor()!=anchor)
     {
     //--- establece el anclaje de la flecha
     arrow.Anchor(anchor);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- utiliza la flecha
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo previamente abierto con la función FileOpen(...) .

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Ejemplo:

```
//--- ejemplo de CChartObjectArrow::Save
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- establece los parámetros del objeto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Flecha",0,TimeCurrent(),price,181))
     {
      //--- error en la creación de la flecha
      printf("Error en la creación de la flecha: %d!",GetLastError());
      //---
      return;
     }
//--- abre el archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Save(file_handle))
        {
         //--- error al guardar el archivo
         printf("Error al guardar el archivo: %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo previamente abierto con la función FileOpen(...).

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Ejemplo:

```
//--- ejemplo de CChartObjectArrow::Load
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- abre el archivo
   file_handle=FileOpen("MiArchivo.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Load(file_handle))
        {
         //--- error al cargar el archivo
         printf("Error al cargar el archivo: %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
//--- utiliza la flecha
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo de objeto (por ejemplo, OBJ_ARROW para CChartObjectArrow)

Ejemplo:

```
//--- ejemplo de CChartObjectArrow::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   //--- obtiene el tipo de flecha
   int type=arrow.Type();
  }
```

© 2000-2025, MetaQuotes Ltd.

Flechas con código fijo

Las clases "Flechas con código fijo" son clases que simplifican el acceso a las propiedades de los siguientes objetos gráficos:

Nombre de la clase	Nombre del objeto flecha
CChartObjectArrowCheck	"Comprobación de la flecha"
CChartObjectArrowDown	"Flecha hacia abajo"
CChartObjectArrowUp	"Flecha hacia arriba"
CChartObjectArrowStop	"Parada de la flecha"
CChartObjectArrowThumbDown	"Bien" ("Dedo pulgar hacia arriba")
CChartObjectArrowThumbUp	"Mal" ("Dedo pulgar hacia abajo")
CChartObjectArrowLeftPrice	Flecha "Precio izquierdo"
CChartObjectArrowRightPrice	Flecha "Precio derecho"

Descripción

Las clases "Flechas con código fijo" proporcionan acceso a las propiedades del objeto.

Declaraciones

```
class CChartObjectArrowCheck      : public CChartObjectArrow;
class CChartObjectArrowDown       : public CChartObjectArrow;
class CChartObjectArrowUp         : public CChartObjectArrow;
class CChartObjectArrowStop       : public CChartObjectArrow;
class CChartObjectArrowThumbDown  : public CChartObjectArrow;
class CChartObjectArrowThumbUp    : public CChartObjectArrow;
class CChartObjectArrowLeftPrice  : public CChartObjectArrow;
class CChartObjectArrowRightPrice : public CChartObjectArrow;
```

Título

```
<ChartObjects\ChartObjectsArrows.mqh>
```

Métodos de la clase

Creación	
Create	Crea el objeto gráfico especificado
Propiedades	
ArrowCode	"Comprobante" del método de cambio de código
Entrada/salida	
virtual Type	Método virtual de identificación

© 2000-2025, MetaQuotes Ltd.

Ver también

Tipos de objetos, Modos de enlace de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Flecha con código fijo".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time,          // Hora
   double    price          // Precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Nombre único del objeto a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time

[in] Coordenada temporal.

price

[in] Coordenada de precio.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

Ejemplo:

```
//--- ejemplo de CChartObjectArrowCheck::Create
//--- ejemplo de CChartObjectArrowDown::Create
//--- ejemplo de CChartObjectArrowUp::Create
//--- ejemplo de CChartObjectArrowStop::Create
//--- ejemplo de CChartObjectArrowThumbDown::Create
//--- ejemplo de CChartObjectArrowThumbUp::Create
//--- ejemplo de CChartObjectArrowLeftPrice::Create
//--- ejemplo de CChartObjectArrowRightPrice::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- por ejemplo, tomar CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- establece los parámetros del objeto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- error en la creación de la flecha
       printf("Error en la creación de la flecha: %d!",GetLastError());
       //---
       return;
      }
//--- utiliza la flecha
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrowCode

Prohíbe los cambios en el código de la flecha.

```
bool  ArrowCode(
   char  code      // valor del código
   )
```

Parámetros

code

[in] cualquier valor

Valor devuelto

Siempre false.

Ejemplo:

```
//--- ejemplo de CChartObjectArrowCheck::ArrowCode
//--- ejemplo de CChartObjectArrowDown::ArrowCode
//--- ejemplo de CChartObjectArrowUp::ArrowCode
//--- ejemplo de CChartObjectArrowStop::ArrowCode
//--- ejemplo de CChartObjectArrowThumbDown::ArrowCode
//--- ejemplo de CChartObjectArrowThumbUp::ArrowCode
//--- ejemplo de CChartObjectArrowLeftPrice::ArrowCode
//--- ejemplo de CChartObjectArrowRightPrice::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- por ejemplo, tomar CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- establece los parámetros del objeto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
     {
      //--- error en la creación de la flecha
      printf("Error en la creación de la flecha: %d!",GetLastError());
      //---
      return;
     }
//--- establece el código de la flecha
   if(!arrow.ArrowCode(181))
     {
      //--- no es un error
      printf("El código de la flecha no se puede cambiar");
     }
//--- utiliza la flecha
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo de objeto gráfico

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo de objeto (OBJ_ARROW_CHECK para CChartObjectArrowCheck, OBJ_ARROW_DOWN para CChartObjectArrowDown, OBJ_ARROW_UP para CChartObjectArrowUp, OBJ_ARROW_STOP para CChartObjectArrowStop, OBJ_ARROW_THUMB_DOWN para CChartObjectArrowThumbDown, OBJ_ARROW_THUMB_UP para CChartObjectArrowThumbUp, OBJ_ARROW_LEFT_PRICE para CChartObjectArrowLeftPrice, OBJ_ARROW_RIGHT_PRICE para CChartObjectArrowRightPrice).

Ejemplo:

```
//--- ejemplo de CChartObjectArrowCheck::Type
//--- ejemplo de CChartObjectArrowDown::Type
//--- ejemplo de CChartObjectArrowUp::Type
//--- ejemplo de CChartObjectArrowStop::Type
//--- ejemplo de CChartObjectArrowThumbDown::Type
//--- ejemplo de CChartObjectArrowThumbUp::Type
//--- ejemplo de CChartObjectArrowLeftPrice::Type
//--- ejemplo de CChartObjectArrowRightPrice::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- por ejemplo, tomar CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- obtener el tipo de flecha
   int type=arrow.Type();
  }
```

© 2000-2025, MetaQuotes Ltd.

Object Controls

Grupo de objetos gráficos "Controles de objeto".

Esta sección contiene los detalles técnicos para trabajar con el grupo de clases de objetos gráficos llamado "Controles de objeto", así como una descripción de los componentes relevantes de la biblioteca estándar de MQL5.

Nombre de la clase	Objeto
CChartObjectText	Objeto gráfico "Texto"
CChartObjectLabel	Objeto gráfico "Etiqueta de texto"
CChartObjectEdit	Objeto gráfico "Editar"
CChartObjectButton	Objeto gráfico "Botón"
CChartObjectSubChart	Objeto gráfico "Gráfico"
CChartObjectBitmap	Objeto gráfico "Bitmap"
CChartObjectBmpLabel	Objeto gráfico "Etiqueta Bitmap"
CChartObjectRectLabel	Objeto gráfico "Etiqueta de rectángulo"

Ver también

Tipos de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

CChartObjectText

La clase CChartObjectText facilita el acceso a las propiedades del objeto gráfico "Texto".

Descripción

La clase CChartObjectText proporciona acceso a las propiedades del objeto "Texto".

Declaración

```
class CChartObjectText : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Jerarquía de herencia

CObject

CChartObject

CChartObjectText

Descendientes directos

CChartObjectLabel

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Texto"
Propiedades	
Angle	Obtiene/Establece la propiedad "Ángulo"
Font	Obtiene/Establece la propiedad "Fuente"
FontSize	Obtiene/Establece la propiedad "Tamaño de la fuente"
Anchor	Obtiene/Establece la propiedad "Anclaje"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Clases derivadas:

- CChartObjectLabel

Ver también

Tipos de objetos, Propiedades de objetos, Modos de enlace de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Texto".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time,          // Coordenada temporal
   double    price          // Coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time

[in] Coordenada temporal del punto de anclaje.

price

[in] Coordenada del precio del punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Angle (Método Get)

Obtiene el valor de la propiedad "Ángulo".

```
double  Angle() const
```

Valor devuelto

Valor de la propiedad "Ángulo" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

Angle (Método Set)

Establece el nuevo valor de la propiedad "Ángulo".

```
bool  Angle(
   double  angle      // ángulo nuevo
   )
```

Parámetros

angle

[in] Nuevo valor de la propiedad "Ángulo".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Font (Método Get)

Obtiene el valor de la propiedad "Fuente".

```
string  Font() const
```

Valor devuelto

Valor de la propiedad "Fuente" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve "".

Font (Método Set)

Establece el nuevo valor de la propiedad "Fuente".

```
bool  Font(
   string  font      // nueva fuente
   )
```

Parámetros

font

[in] Nuevo valor de la propiedad "Fuente".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

FontSize (Método Get)

Obtiene el valor de la propiedad "Tamaño de la fuente".

```
int  FontSize() const
```

Valor devuelto

Valor de la propiedad "Tamaño de la fuente" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

FontSize (Método Set)

Establece el nuevo valor de la propiedad "Tamaño de la fuente".

```
bool  FontSize(
   int  size      // nuevo tamaño de la fuente
   )
```

Parámetros

size

[in] Nuevo valor de la propiedad "Fuente".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Anchor (Método Get)

Obtiene el valor de la propiedad "Anclaje".

```
ENUM_ANCHOR_POINT  Anchor() const
```

Valor devuelto

Valor de la propiedad "Anclaje" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve WRONG_VALUE.

Anchor (Método Set)

Establece el valor nuevo de la propiedad "Anclaje".

```
bool  Anchor(
   ENUM_ANCHOR_POINT  anchor      // valor nuevo
   )
```

Parámetros

anchor

[in] Nuevo valor de la propiedad "Anclaje".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_TEXT para CChartObjectText).

© 2000-2025, MetaQuotes Ltd.

CChartObjectLabel

La clase CChartObjectLabel facilita el acceso a las propiedades del objeto gráfico "Etiqueta".

Descripción

La clase CChartObjectLabel proporciona acceso a las propiedades del objeto "Etiqueta".

Declaración

```
class CChartObjectLabel : public CChartObjectText
```

Título

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel

Descendientes directos

CChartObjectEdit, CChartObjectRectLabel

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Etiqueta"
Propiedades	
X_Distance	Obtiene/Establece la propiedad "X_Distance"
Y_Distance	Obtiene/Establece la propiedad "Y_Distance"
X_Size	Obtiene/Establece la propiedad "X_Size"
Y_Size	Obtiene/Establece la propiedad "Y_Size"
Corner	Obtiene/Establece la propiedad "Esquina"
Time	"Comprobante" del cambio en la coordenada temporal
Price	"Comprobante" del cambio en la coordenada del precio
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Ver también

Tipos de objetos, Propiedades de objetos, Ángulo del gráfico, Modos de enlace de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Etiqueta".

```
bool  Create(
   long    chart_id,      // Identificador del gráfico
   string  name,          // Nombre del objeto
   int     window,        // Ventana del gráfico
   int     X,             // Coordenada X
   int     Y              // Coordenada Y
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

X

[in] Coordenada X.

Y

[in] Coordenada Y.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

X_Distance (Método Get)

Obtiene el valor de la propiedad "X_Distance".

```
int  X_Distance() const
```

Valor devuelto

Valor de la propiedad "X_Distance" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

X_Distance (Método Set)

Establece el nuevo valor de la propiedad "X_Distance".

```
bool  X_Distance(
   int  X      // valor nuevo
   )
```

Parámetros

X

[in] Valor nuevo de la propiedad "X_Distance".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Y_Distance (Método Get)

Obtiene el valor de la propiedad "Y_Distance".

```
int  Y_Distance() const
```

Valor devuelto

Valor de la propiedad "Y_Distance" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

Y_Distance (Método Set)

Establece el nuevo valor de la propiedad "Y_Distance".

```
bool  Y_Distance(
   int  Y      // valor nuevo
   )
```

Parámetros

Y

[in] Valor nuevo de la propiedad "Y_Distance".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

X_Size

Obtiene el valor de la propiedad "X_Size".

```
int  X_Size() const
```

Valor devuelto

Valor de la propiedad "X_Size" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

Y_Size

Obtiene el valor de la propiedad "Y_Size".

```
int  Y_Size() const
```

Valor devuelto

Valor de la propiedad "Y_Size" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

Corner (Método Get)

Obtiene el valor de la propiedad "Corner".

```
ENUM_BASE_CORNER  Corner() const
```

Valor devuelto

Valor de la propiedad "Corner" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve WRONG_VALUE.

Corner (Método Set)

Establece el nuevo valor de la propiedad "Esquina".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // nuevo valor
   )
```

Parámetros

corner

[in] Nuevo valor de la propiedad "Esquina".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Time

Impide los cambios en la coordenada temporal.

```
bool  Time(
   datetime  time      // cualquier valor
   )
```

Parámetros

time

[in] Cualquier valor de tipo datetime.

Valor devuelto

Siempre false.

© 2000-2025, MetaQuotes Ltd.

Price

Impide los cambios en la coordenada del precio.

```
bool  Price(
   double  price      // cualquier valor
   )
```

Parámetros

price

[in] Cualquier valor de tipo double.

Valor devuelto

Siempre false.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_LABEL para CChartObjectLabel).

© 2000-2025, MetaQuotes Ltd.

CChartObjectEdit

La clase CChartObjectEdit facilita el acceso a las propiedades del objeto gráfico "Editar".

Descripción

La clase CChartObjectEdit proporciona acceso a las propiedades del objeto "Editar".

Declaración

```
class CChartObjectEdit : public CChartObjectLabel
```

Título

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Jerarquía de herencia

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectEdit

Descendientes directos

CChartObjectButton

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Editar"
Propiedades	
TextAlign	Obtiene/Establece la propiedad "Alineación del texto"
X_Size	Obtiene la propiedad "Tamaño X"
Y_Size	Obtiene la propiedad "Tamaño Y"
BackColor	Obtiene/Establece la propiedad "Color de fondo"
BorderColor	Obtiene/Establece la propiedad "Color del borde"
ReadOnly	Obtiene/Establece la propiedad "Solo lectura"
Angle	Obtiene/Establece la propiedad "Ángulo"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo

© 2000-2025, MetaQuotes Ltd.

Create	
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Métodos heredados de la clase CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Ver también

Tipos de objetos, Propiedades de objetos, Ángulo del gráfico, Modos de enlace de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Editar".

```
bool  Create(
   long    chart_id,      // Identificador del gráfico
   string  name,          // Nombre del objeto
   int     window,        // Ventana del gráfico
   int     X,             // Coordenada X
   int     Y,             // Coordenada Y
   int     sizeX,         // Tamaño X
   int     sizeY          // Tamaño Y
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

X

[in] Coordenada X.

Y

[in] Coordenada Y.

sizeX

[in] Tamaño X.

sizeY

[in] Tamaño Y.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

TextAlign (Método Get)

Obtiene el valor de la propiedad "Alineación del texto" (modo alineación de texto).

```
ENUM_ALIGN_MODE  TextAlign() const
```

Valor devuelto

Valor de la propiedad "Alineación del texto" del objeto, asignado a la instancia de la clase.

TextAlign (Método Set)

Establece el valor de la propiedad "Alineación del texto" (modo alineación de texto).

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // valor nuevo
   )
```

Parámetros

align

[in] Valor nuevo de la propiedad "Alineación del texto".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

X_Size

Establece el nuevo valor de la propiedad "X_Size".

```
bool  X_Size(
   int  size      // nuevo tamaño
   )
```

Parámetros

size

[in] Nuevo valor de la propiedad "X_Size".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Y_Size

Establece el nuevo valor de la propiedad "Y_Size".

```
bool  Y_Size(
   int  size      // nuevo tamaño
   )
```

Parámetros

size

[in] Nuevo valor de la propiedad "Y_Size".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

BackColor (Método Get)

Obtiene el valor de la propiedad "Color de fondo".

```
color  BackColor() const
```

Valor devuelto

Valor de la propiedad "Color de fondo" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve CLR_NONE.

BackColor (Método Set)

Establece el nuevo valor de la propiedad "Color de fondo".

```
bool  BackColor(
   color  new_color      // nuevo color de fondo
   )
```

Parámetros

new_color

[in] Valor nuevo de la propiedad "Color de fondo".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

BorderColor (Método Get)

Obtiene el valor de la propiedad "Color del borde".

```
color  BorderColor() const
```

Valor devuelto

Valor de la propiedad "Color del borde" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve CLR_NONE.

BorderColor (Método Set)

Establece el nuevo valor de la propiedad "Color del borde".

```
bool  BorderColor(
   color  new_color      // nuevo color del borde
   )
```

Parámetros

new_color

[in] Valor nuevo de la propiedad "Color del borde".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

ReadOnly (Método Get)

Obtiene el valor de la propiedad "Solo lectura".

```
bool  ReadOnly() const
```

Valor devuelto

Valor de la propiedad "Solo lectura" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

ReadOnly (Método Set)

Establece el nuevo valor de la propiedad "Solo lectura".

```
bool  ReadOnly(
   const  bool  flag      // valor nuevo
   )
```

Parámetros

flag

[in] Nuevo valor de la propiedad "Solo lectura".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Angle

Impide los cambios en la propiedad "Ángulo".

```
bool  Angle(
   double  angle      // cualquier valor
   )
```

Parámetros

angle

[in] Cualquier valor de tipo double.

Valor devuelto

Siempre false.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_EDIT para CChartObjectEdit).

© 2000-2025, MetaQuotes Ltd.

CChartObjectButton

La clase CChartObjectButton facilita el acceso a las propiedades del objeto gráfico "Botón".

Descripción

La clase CChartObjectButton proporciona acceso a las propiedades del objeto "Botón".

Declaración

```
class CChartObjectButton : public CChartObjectEdit
```

Título

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Jerarquía de herencia

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectEdit
CChartObjectButton

Descendientes directos

CChartObjectPanel

Métodos de la clase

Create	
Create	Clase que hereda de CChartObjectEdit
Propiedades	
State	Obtiene/Establece el estado del botón (Presionado/No presionado)
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Métodos heredados de la clase CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Métodos heredados de la clase CChartObjectEdit

X_Size, Y_Size, BackColor, BackColor, BorderColor, BorderColor, ReadOnly, ReadOnly, TextAlign, TextAlign, Angle, Create

Ver también

Tipos de objetos, Propiedades de objetos, Ángulo del gráfico, Modos de enlace de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

State (Método Get)

Obtiene el valor de la propiedad "Estado".

```
bool  State() const
```

Valor devuelto

Valor de la propiedad "Estado" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

State (Método Set)

Establece el nuevo valor de la propiedad "Estado".

```
bool  State(
   bool  state      // nuevo valor del estado
   )
```

Parámetros

X

[in] Nuevo valor de la propiedad "Estado".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_BUTTON para CChartObjectButton).

© 2000-2025, MetaQuotes Ltd.

CChartObjectSubChart

La clase CChartObjectSubChart facilita el acceso a las propiedades del objeto gráfico "Chart".

Descripción

La clase CChartObjectSubChart proporciona acceso a las propiedades del objeto "Chart".

Declaración

```
class CChartObjectSubChart : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectSubChart.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectSubChart

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Chart"
Propiedades	
X_Distance	Obtiene/Establece la propiedad "X_Distance"
Y_Distance	Obtiene/Establece la propiedad "Y_Distance"
Corner	Obtiene/Establece la propiedad "Esquina"
X_Size	Obtiene/Establece la propiedad "X_Size"
Y_Size	Obtiene/Establece la propiedad "Y_Size"
Symbol	Obtiene/Establece la propiedad "Símbolo"
Period	Obtiene/Establece la propiedad "Periodo"
Scale	Obtiene/Establece la propiedad "Escala"
DateScale	Obtiene/Establece la propiedad "Mostrar escala de la fecha"
PriceScale	Obtiene/Establece la propiedad "Mostrar escala del precio"
Time	"Comprobante" del cambio de la coordenada temporal
Price	"Comprobante" del cambio de la coordenada del precio
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo

© 2000-2025, MetaQuotes Ltd.

Create	
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Ver también

Tipos de objetos, Propiedades de objetos, Ángulo del gráfico, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "SubChart".

```
bool  Create(
   long    chart_id,      // Identificador del gráfico
   string  name,          // Nombre del objeto
   int     window,        // Ventana del gráfico
   int     X,             // Coordenada X
   int     Y,             // Coordenada Y
   int     sizeX,         // Tamaño X
   int     sizeY          // Tamaño Y
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

X

[in] Coordenada X.

Y

[in] Coordenada Y.

sizeX

[in] Tamaño X.

sizeY

[in] Tamaño Y.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

X_Distance (Método Get)

Obtiene el valor de la propiedad "X_Distance".

```
int  X_Distance() const
```

Valor devuelto

Valor de la propiedad "X_Distance" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

X_Distance (Método Set)

Establece el nuevo valor de la propiedad "X_Distance".

```
bool  X_Distance(
   int  X      // valor nuevo
   )
```

Parámetros

X

[in] Valor nuevo de la propiedad "X_Distance".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Y_Distance (Método Get)

Obtiene el valor de la propiedad "Y_Distance".

```
int  Y_Distance() const
```

Valor devuelto

Valor de la propiedad "Y_Distance" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

Y_Distance (Método Set)

Establece el nuevo valor de la propiedad "Y_Distance".

```
bool  Y_Distance(
   int  Y      // valor nuevo
   )
```

Parámetros

Y

[in] Valor nuevo de la propiedad "Y_Distance".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Corner (Método Get)

Obtiene el valor de la propiedad "Corner".

```
ENUM_BASE_CORNER Corner() const
```

Valor devuelto

Valor de la propiedad "Corner" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve WRONG_VALUE.

Corner (Método Set)

Establece el nuevo valor de la propiedad "Esquina".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // nuevo valor
   )
```

Parámetros

corner

[in] Nuevo valor de la propiedad "Esquina".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

X_Size (Método Get)

Obtiene el valor de la propiedad "X_Size".

```
int  X_Size() const
```

Valor devuelto

Valor de la propiedad "X_Size" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

X_Size (Método Set)

Establece el nuevo valor de la propiedad "X_Size".

```
bool  X_Size(
   int  X      // valor nuevo
   )
```

Parámetros

X

[in] Nuevo valor de la propiedad "X_Size".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Y_Size (Método Get)

Obtiene el valor de la propiedad "Y_Size".

```
int  Y_Size() const
```

Valor devuelto

Valor de la propiedad "Y_Size" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

Y_Size (Método Set)

Establece el nuevo valor de la propiedad "Y_Size".

```
bool  Y_Size(
   int  Y      // valor nuevo
   )
```

Parámetros

Y

[in] Nuevo valor de la propiedad "Y_Size".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Symbol (Método Get)

Obtiene el valor de la propiedad "Símbolo".

```
string  Symbol() const
```

Valor devuelto

Valor de la propiedad "Símbolo" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve "".

Symbol (Método Set)

Establece el nuevo valor de la propiedad "Símbolo".

```
bool  Symbol(
   string  symbol      // símbolo nuevo
   )
```

Parámetros

symbol

[in] Nuevo valor de la propiedad "Símbolo".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Period (Método Get)

Obtiene el valor de la propiedad "Periodo".

```
int  Period() const
```

Valor devuelto

Valor de la propiedad "Period" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

Period (Método Set)

Establece el nuevo valor de la propiedad "Periodo".

```
bool  Period(
   int  period      // periodo nuevo
   )
```

Parámetros

period

[in] Nuevo valor de la propiedad "Periodo".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Scale (Método Get)

Obtiene el valor de la propiedad "Escala".

```
double  Scale() const
```

Valor devuelto

Valor de la propiedad "Escala" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

Scale (Método Set)

Establece el nuevo valor de la propiedad "Escala".

```
bool  Scale(
   double  scale      // nueva escala
   )
```

Parámetros

scale

[in] Nuevo valor de la propiedad "Escala".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

DateScale (Método Get)

Obtiene el valor de la propiedad "DateScale".

```
bool  DateScale() const
```

Valor devuelto

Valor de la propiedad "DateScale" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

DateScale (Método Set)

Establece el nuevo valor de la propiedad "DateScale".

```
bool  DateScale(
   bool  scale      // valor nuevo
   )
```

Parámetros

scale

[in] Valor nuevo de la propiedad "DateScale".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

PriceScale (Método Get)

Obtiene el valor de la propiedad "Escala del precio".

```
bool  PriceScale() const
```

Valor devuelto

Valor de la propiedad "Escala del precio" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

PriceScale (Método Set)

Establece el nuevo valor de la propiedad "Escala del precio".

```
bool  PriceScale(
   bool  scale      // valor nuevo
   )
```

Parámetros

scale

[in] Nuevo valor de la propiedad "Escala del precio".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Time

Impide los cambios en la coordenada temporal.

```
bool  Time(
   datetime  time      // cualquier valor
   )
```

Parámetros

time

[in] Cualquier valor de tipo datetime.

Valor devuelto

Siempre false.

© 2000-2025, MetaQuotes Ltd.

Price

Impide los cambios en la coordenada del precio.

```
bool  Price(
   double  price      // cualquier valor
   )
```

Parámetros

price

[in] Cualquier valor de tipo double.

Valor devuelto

Siempre false.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_CHART para CChartObjectSubChart).

© 2000-2025, MetaQuotes Ltd.

CChartObjectBitmap

La clase CChartObjectBitmap facilita el acceso a las propiedades del objeto gráfico "Bitmap".

Descripción

La clase CChartObjectBitmap proporciona acceso a las propiedades del objeto "Bitmap".

Declaración

```
class CChartObjectBitmap : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectBitmap

Métodos de la clase

Create	
Create	Crea el objeto gráfico "Bitmap"
Propiedades	
BmpFile	Obtiene/Establece la propiedad "Nombre de archivo BMP"
X_Offset	Obtiene/Establece la propiedad "X_Offset"
Y_Offset	Obtiene/Establece la propiedad "Y_Offset"
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Ver también

Tipos de objetos, Propiedades de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "Bitmap".

```
bool  Create(
   long      chart_id,      // Identificador del gráfico
   string    name,          // Nombre del objeto
   int       window,        // Ventana del gráfico
   datetime  time,          // Coordenada temporal
   double    price          // Coordenada del precio
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Un nombre único, para el objeto que se va a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

time

[in] Coordenada temporal del punto de anclaje.

price

[in] Coordenada del precio del punto de anclaje.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

BmpFile (Método Get)

Obtiene el valor de la propiedad "BmpFile".

```
string  BmpFile() const
```

Valor devuelto

Valor de la propiedad "BmpFile" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

BmpFile (Método Set)

Establece el nuevo valor de la propiedad "BmpFile".

```
bool  BmpFile(
   string  name      // nombre de archivo nuevo
   )
```

Parámetros

name

[in] Nuevo valor de la propiedad "BmpFile".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

X_Offset (Método Get)

Obtiene el valor de la propiedad "X_Offset" (la coordenada X de la esquina superior izquierda del área rectangular visible de los objetos gráficos).

```
int  X_Offset() const
```

Valor devuelto

Valor de la propiedad "X_Offset" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

X_Offset (Método Set)

Establece el nuevo valor de la propiedad "X_Offset" (la coordenada X de la esquina superior izquierda del área rectangular visible de los objetos gráficos). El valor se establece en píxeles con respecto a la esquina superior izquierda de la imagen original.

```
bool  X_Offset(
   int  X      // valor nuevo
   )
```

Parámetros

X

[in] Valor nuevo de la propiedad "X_Offset".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Y_Offset (Método Get)

Gets the value of "Y_Offset" property (la coordenada Y de la esquina superior izquierda del área rectangular visible de los objetos gráficos).

```
int  Y_Offset() const
```

Valor devuelto

Valor de la propiedad "Y_Offset" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

Y_Offset (Método Set)

Establece el nuevo valor de la propiedad "Y_Offset" (la coordenada Y de la esquina superior izquierda del área rectangular visible de los objetos gráficos). El valor se establece en píxeles con respecto a la esquina superior izquierda de la imagen original.

```
bool  Y_Offset(
   int  Y      // valor nuevo
   )
```

Parámetros

Y

[in] Nuevo valor de la propiedad "Y_Offset".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_BITMAP para CChartObjectBitmap).

© 2000-2025, MetaQuotes Ltd.

CChartObjectBmpLabel

La clase CChartObjectBmpLabel facilita el acceso a las propiedades del objeto gráfico "Etiqueta Bitmap".

Descripción

La clase CChartObjectBmpLabel proporciona acceso a las propiedades del objeto "Etiqueta Bitmap".

Declaración

```
class CChartObjectBmpLabel : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectBmpLabel

Métodos de la clase

Create	
Create	Crea el objeto gráfico "BmpLabel"
Propiedades	
X_Distance	Obtiene/Establece la propiedad "X_Distance"
Y_Distance	Obtiene/Establece la propiedad "Y_Distance"
X_Offset	Obtiene/Establece la propiedad "X_Offset"
Y_Offset	Obtiene/Establece la propiedad "Y_Offset"
Corner	Obtiene/Establece la propiedad "Esquina"
X_Size	Obtiene/Establece la propiedad "X_Size"
Y_Size	Obtiene/Establece la propiedad "Y_Size"
BmpFileOn	Obtiene/Establece la propiedad "BmpFileOn", estado del botón: presionado (On)
BmpFileOff	Obtiene/Establece la propiedad "BmpFileOff", estado del botón: no presionado (Off)
State	Obtiene/Establece la propiedad "Estado del botón" (Presionado/No presionado)
Time	"Comprobante" del cambio de la coordenada temporal

© 2000-2025, MetaQuotes Ltd.

Create	
Price	"Comprobante" del cambio de la coordenada del precio
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Ver también

Tipos de objetos, Propiedades de objetos, Ángulo del gráfico, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "BmpLabel".

```
bool  Create(
   long    chart_id,      // Identificador del gráfico
   string  name,          // Nombre del objeto
   int     window,        // Ventana del gráfico
   int     X,             // Coordenada X
   int     Y              // Coordenada Y
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Nombre único del objeto a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

X

[in] Coordenada X.

Y

[in] Coordenada Y.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

X_Distance (Método Get)

Obtiene el valor de la propiedad "X_Distance".

```
int  X_Distance() const
```

Valor devuelto

Valor de la propiedad "X_Distance" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

X_Distance (Método Set)

Establece el nuevo valor de la propiedad "X_Distance".

```
bool  X_Distance(
   int  X      // valor nuevo
   )
```

Parámetros

X

[in] Valor nuevo de la propiedad "X_Distance".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Y_Distance (Método Get)

Obtiene el valor de la propiedad "Y_Distance".

```
int  Y_Distance() const
```

Valor devuelto

Valor de la propiedad "Y_Distance" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

Y_Distance (Método Set)

Establece el nuevo valor de la propiedad "Y_Distance".

```
bool  Y_Distance(
   int  Y      // valor nuevo
   )
```

Parámetros

Y

[in] Valor nuevo de la propiedad "Y_Distance".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

X_Offset (Método Get)

Obtiene el valor de la propiedad "X_Offset" (la coordenada X de la esquina superior izquierda del área rectangular visible de los objetos gráficos).

```
int  X_Offset() const
```

Valor devuelto

Valor de la propiedad "X_Offset" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

X_Offset (Método Set)

Establece el nuevo valor de la propiedad "X_Offset" (la coordenada X de la esquina superior izquierda del área rectangular visible de los objetos gráficos). El valor se establece en píxeles con respecto a la esquina superior izquierda de la imagen original.

```
bool  X_Offset(
   int  X      // valor nuevo
   )
```

Parámetros

X

[in] Valor nuevo de la propiedad "X_Offset".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Y_Offset (Método Get)

Gets the value of "Y_Offset" property (la coordenada Y de la esquina superior izquierda del área rectangular visible de los objetos gráficos).

```
int  Y_Offset() const
```

Valor devuelto

Valor de la propiedad "Y_Offset" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

Y_Offset (Método Set)

Establece el nuevo valor de la propiedad "Y_Offset" (la coordenada Y de la esquina superior izquierda del área rectangular visible de los objetos gráficos). El valor se establece en píxeles con respecto a la esquina superior izquierda de la imagen original.

```
bool  Y_Offset(
   int  Y      // valor nuevo
   )
```

Parámetros

Y

[in] Nuevo valor de la propiedad "Y_Offset".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Corner (Método Get)

Obtiene el valor de la propiedad "Corner".

```
ENUM_BASE_CORNER Corner() const
```

Valor devuelto

Valor de la propiedad "Corner" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve WRONG_VALUE.

Corner (Método Set)

Establece el nuevo valor de la propiedad "Esquina".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // nuevo valor
   )
```

Parámetros

corner

[in] Nuevo valor de la propiedad "Esquina".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

X_Size

Obtiene el valor de la propiedad "X_Size".

```
int  X_Size() const
```

Valor devuelto

Valor de la propiedad "X_Size" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

Y_Size

Obtiene el valor de la propiedad "Y_Size".

```
int  Y_Size() const
```

Valor devuelto

Valor de la propiedad "Y_Size" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

BmpFileOn (Método Get)

Obtiene el valor de la propiedad "BmpFileOn".

```
string  BmpFileOn() const
```

Valor devuelto

Valor de la propiedad "BmpFileOn" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve "".

BmpFileOn (Método Set)

Establece el nuevo valor de la propiedad "BmpFileOn".

```
bool  BmpFileOn(
   string  name      // nombre del archivo
   )
```

Parámetros

name

[in] Valor nuevo de la propiedad "BmpFileOn".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

BmpFileOff (Método Get)

Obtiene el valor de la propiedad "BmpFileOff".

```
string  BmpFileOff() const
```

Valor devuelto

Valor de la propiedad "BmpFileOff" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve "".

BmpFileOff (Método Set)

Establece el nuevo valor de la propiedad "BmpFileOff".

```
bool  BmpFileOff(
   string  name      // nombre del archivo
   )
```

Parámetros

name

[in] Valor nuevo de la propiedad "BmpFileOff".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

State (Método Get)

Obtiene el valor de la propiedad "Estado".

```
bool  State() const
```

Valor devuelto

Valor de la propiedad "Estado" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

State (Método Set)

Establece el nuevo valor de la propiedad "Estado".

```
bool  State(
   bool  state      // nuevo valor del estado
   )
```

Parámetros

state

[in] Nuevo valor de la propiedad "Estado".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Time

Impide los cambios en la coordenada temporal.

```
bool  Time(
   datetime  time      // cualquier valor
   )
```

Parámetros

time

[in] Cualquier valor de tipo datetime.

Valor devuelto

Siempre false.

© 2000-2025, MetaQuotes Ltd.

Price

Impide los cambios en la coordenada del precio.

```
bool  Price(
   double  price      // cualquier valor
   )
```

Parámetros

price

[in] Cualquier valor de tipo double.

Valor devuelto

Siempre false.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_BITMAP_LABEL para CChartObjectBmpLabel).

© 2000-2025, MetaQuotes Ltd.

CChartObjectRectLabel

La clase CChartObjectRectLabel facilita el acceso a las propiedades del objeto gráfico "Rectangle Label".

Descripción

La clase CChartObjectRectLabel proporciona acceso a las propiedades del objeto "Rectangle Label".

Declaración

```
class CChartObjectRectLabel : public CChartObjectLabel
```

Título

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Jerarquía de herencia

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectRectLabel

Métodos de la clase

Create	
Create	Crea el objeto gráfico "RectLabel"
Propiedades	
X_Size	Establece el tamaño horizontal
Y_Size	Establece el tamaño vertical
BackColor	Obtiene/Establece el color de fondo
Angle	Método comprobante
BorderType	Obtiene/Establece el tipo de borde
Entrada/salida	
virtual Save	Método virtual para escribir en un archivo
virtual Load	Método virtual para leer un archivo
virtual Type	Método virtual de identificación

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos heredados de la clase CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Métodos heredados de la clase CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Ver también

Tipos de objetos, Propiedades de objetos, Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Create

Crea el objeto gráfico "CChartObjectRectLabel".

```
bool  Create(
   long    chart_id,      // Identificador del gráfico
   string  name,          // Nombre del objeto
   int     window,        // Ventana del gráfico
   int     X,             // Coordenada X
   int     Y,             // Coordenada Y
   int     sizeX,         // Tamaño horizontal
   int     sizeY          // Tamaño vertical
   )
```

Parámetros

chart_id

[in] Identificador del gráfico (0 - gráfico actual).

name

[in] Nombre único del objeto a crear.

window

[in] Número de la ventana del gráfico (0 - ventana base).

X

[in] Coordenada X.

Y

[in] Coordenada Y.

sizeX

[in] Tamaño horizontal.

sizeY

[in] Tamaño vertical.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

X_Size

Establece el valor de la propiedad "X_Size".

```
bool  X_Size(
   int  size      // Tamaño horizontal
   )
```

Parámetros

size

[in] Nuevo tamaño horizontal.

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no cambia.

Nota

Para obtener las propiedades "X_Size" y "Y_Size" utilice los métodos X_Size y Y_Size disponibles en la clase padre CChartObjectLabel.

© 2000-2025, MetaQuotes Ltd.

Y_Size

Establece el valor de la propiedad "Y_Size".

```
bool  Y_Size(
   int  size      // Tamaño vertical
   )
```

Parámetros

size

[in] Nuevo tamaño vertical.

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no cambia.

Nota

Para obtener las propiedades "X_Size" y "Y_Size" utilice los métodos X_Size y Y_Size disponibles en la clase padre CChartObjectLabel.

© 2000-2025, MetaQuotes Ltd.

BackColor

Obtiene el color de fondo.

```
color  BackColor() const
```

Valor devuelto

Color de fondo del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

BackColor

Establece el color de fondo.

```
bool  BackColor(
   color  new_color      // Color nuevo
   )
```

Parámetros

new_color

[in] Nuevo color de fondo.

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Angle

Método comprobante.

```
bool  Angle(
   double  angle      // cualquier valor
   )
```

Parámetros

angle

[in] Cualquier valor de tipo double.

Valor devuelto

Siempre false.

© 2000-2025, MetaQuotes Ltd.

BorderType

Obtiene el tipo de borde.

```
int  BorderType() const
```

Valor devuelto

Tipo de borde del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

BorderType

Establece el tipo de borde.

```
bool  BorderType(
   int  type      // Tipo de borde
   )
```

Parámetros

type

[in] Tipo de borde.

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario previamente abierto con la función FileOpen.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador del tipo de objeto gráfico.

```
virtual int  Type() const
```

Valor devuelto

Identificador del tipo de objeto (OBJ_RECTANGLE_LABEL para CChartObjectRectangleLabel).

© 2000-2025, MetaQuotes Ltd.

Gráfico personalizado

En este apartado se presentan los instrumentos para trabajar con los gráficos personalizados.

Su uso simplifica significativamente la construcción de los gráficos personalizados, las figuras y los datos visuales.

Se han desarrollado aparte clases para la creación de objetos gráficos y primitivas para dibujar diferentes tipos de diagramas y curvas. Se han implementado diferentes posibilidades de representación de objetos: cambio del estilo y el color de las líneas, coloreado, trabajo con los datos en el gráfico, etcétera.

Clase	Descripción
CCanvas	Clases para la creación simplificada de figuras personalizadas
CChartCanvas	Clase básica para implementar las clases destinadas al dibujado de gráficos y sus elementos
CHistogramChart	Clase para construir histogramas con columnas
CLineChart	Clase para dibujar curvas
CPieChart	Clase para construir diagramas circulares

© 2000-2025, MetaQuotes Ltd.

CCanvas

La clase CCanvas es la clase para la creación simplificada de dibujos personalizados.

Descripción

La clase CCanvas posibilita la creación de un recurso gráfico (vinculado o no vinculado a un objeto del gráfico) y el dibujado de primitivos gráficos.

Declaración

```
class CCanvas
```

Encabezamiento

```
#include <Canvas\Canvas.mqh>
```

Jerarquía de herencia

CCanvas

Descendientes directos

CChartCanvas, CFlameCanvas

Método de clase por grupos

Creación	
Attach	Vincula el objeto OBJ_BITMAP_LABEL al ejemplar de la clase CCanvas
Create	Crea un recurso gráfico no vinculado a un objeto del gráifico
CreateBitmap	Crea un recurso gráfico vinculado a un objeto del gráfico
CreateBitmapLabel	Crea un recurso gráfico vinculado a un objeto del gráfico
Destroy	Elimina un recurso gráfico
Propiedades	
ChartObjectName	Obtiene el nombre del objeto vinculado del gráfico
ResourceName	Obtiene el nombre del recurso gráfico
Width	Obtiene la anchura del recurso gráfico
Height	Obtiene la altura del recurso gráfico
LineStyleSet	Establece el estilo de la línea
Actualización del objeto en la pantalla	
Update	Representa los cambios en la pantalla

© 2000-2025, MetaQuotes Ltd.

Creación	
Resize	Cambia las dimensiones del recurso gráfico
Borrado/coloreado	
Erase	Borra o colorea con el color indicado
Acceso a los datos	
PixelGet	Obtiene el color del punto con las coordenadas indicadas
PixelSet	Establece el color del punto con las coordenadas indicadas
Dibujado de primitivos	
LineVertical	Dibuja una línea vertical
LineHorizontal	Dibuja una línea horizontal
Line	Dibuja una línea aleatoria
Polyline	Dibuja una línea quebrada
Polygon	Dibuja un polígono
Rectangle	Dibuja un rectángulo
Circle	Dibuja una circunferencia
Triangle	Dibuja un triángulo
Ellipse	Dibuja una elipse
Arc	Dibuja un arco de elipse
Pie	Dibuja un sector de la elipse
Dibujado de primitivos coloreados	
FillRectangle	Dibuja un rectángulo coloreado
FillCircle	Dibuja un círculo coloreado
FillTriangle	Dibuja un triángulo coloreado
FillPolygon	Dibuja un polígono coloreado
FillEllipse	Dibuja una elipse coloreada
Fill	Colorea un zona
Dibujado de primitivos con uso de suavizado	
PixelSetAA	Dibuja un punto
LineAA	Dibuja una línea
PolylineAA	Dibuja una línea quebrada

© 2000-2025, MetaQuotes Ltd.

Creación	
PolygonAA	Dibuja un polígono
TriangleAA	Dibuja un triángulo
CircleAA	Dibuja una circunferencia
EllipseAA	Dibuja una elipse
LineWu	Dibuja una línea
PolylineWu	Dibuja una línea quebrada
PolygonWu	Dibuja un polígono
TriangleWu	Dibuja un triángulo
CircleWu	Dibuja una circunferencia
EllipseWu	Dibuja una elipse
LineThick	Dibuja un segmento de una línea aleatoria con un grosor establecido usando un algoritmo de suavizado
LineThickVertical	Dibuja un segmento vertical de una línea aleatoria con un grosor establecido usando un algoritmo de suavizado
LineThickHorizontal	Dibuja un segmento horizontal de una línea aleatoria con un grosor establecido usando un algoritmo de suavizado
PolygonSmooth	Dibuja un polígono con un grosor establecido usando dos algoritmos de suavizado
PolygonThick	Dibuja un polígono con un grosor establecido usando un algoritmo de suavizado
PolylineSmooth	Dibuja una línea quebrada con un grosor establecido usando dos algoritmos de suavizado
PolylineThick	Dibuja una línea quebrada con un grosor establecido usando un algoritmo de suavizado
Texto	
FontSet	Establece los parámetros de la fuente
FontNameSet	Establece el nombre de la fuente
FontSizeSet	Establece el tamaño de la fuente
FontFlagsSet	Establece las banderas de la fuente
FontAngleSet	Establece el ángulo de inclinación de la fuente
FontGet	Obtiene los parámetros de la fuente
FontNameGet	Obtiene el nombre de la fuente
FontSizeGet	Obtiene el tamaño de la fuente

© 2000-2025, MetaQuotes Ltd.

Creación	
FontFlagsGet	Obtiene las banderas de la fuente
FontAngleGet	Obtiene el ángulo de inclinación de la fuente
TextOut	Muestra el texto
TextWidth	Obtiene la anchura del texto
TextHeight	Obtiene la altura del texto
TextSize	Obtiene las dimensiones del texto
Transparencia	
TransparentLevelSet	Establece el nivel de transpariencia
Entrada/salida	
LoadFromFile	Lee la imagen de un archivo BMP

© 2000-2025, MetaQuotes Ltd.

Attach

Obtiene del objeto OBJ_BITMAP_LABEL un recurso gráfico y lo conecta a un ejemplar de la clase CCanvas.

```
bool  Attach(
   const long          chart_id,                          // identificador del grá
   const string        objname,                           // nombre del objeto
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // método de procesamient
   )
```

Crea un recurso gráfico para el objeto OBJ_BITMAP_LABEL y lo conecta a un ejemplar de la clase CCanvas.

```
bool  Attach(
   const long          chart_id,                          // identificador del grá
   const string        objname,                           // nombre del objeto
   const int           width,                             // anchura de la imagen e
   const int           height,                            // altura de la imagen en
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // método de procesamient
   )
```

Parámetros

chart_id

[out] Identificador del gráfico.

objname

[in] Denominación (nombre) del objeto gráfico.

width

[in] Anchura de la imagen en el recurso.

height

[in] Altura de la imagen en el recurso.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Método de procesamiento del canal alfa. Por defecto, el canal alfa se ignora.

Valor devuelto

true - en el caso de éxito, false - si no se ha logrado conectar el objeto.

© 2000-2025, MetaQuotes Ltd.

Arc

Dibuja un arco de la elipse inscrito en el rectángulo con los ángulos (x1,y1) y (x2,y2). Los límites del arco se cortan con las líneas del centro de la elipse que van a los dos puntos con las coordenadas (x3,y3) y (x4,y4).

```
void  Arc(
   int          x1,      // coordenada X del ángulo superior izquierdo del rectángulo
   int          y1,      // coordenada Y del ángulo superior izquierdo del rectángulo
   int          x2,      // coordenada X del ángulo inferior derecho del rectángulo
   int          y2,      // coordenada Y del ángulo inferior derecho del rectángulo
   int          x3,      // coordenada X del primer punto para encontrar el límite del
   int          y3,      // coordenada Y del primer punto para encontrar el límite del
   int          x4,      // coordenada X del segundo punto para encontrar el límite del
   int          y4,      // coordenada Y del segundo punto para encontrar el límite del
   const uint   clr      // color
   );
```

Parámetros

x1

[in] Coordenada X del ángulo superior izquierdo que define el rectángulo.

y1

[in] Coordenada Y del ángulo superior izquierdo que define el rectángulo.

x2

[in] Coordenada X del ángulo inferior derecho que define el rectángulo.

y2

[in] Coordenada Y del ángulo inferior derecho que define el rectángulo.

x3

[in] Coordenada X del primer punto hacia el que se traza la línea desde el centro del rectángulo para obtener el límite del arco.

y3

[in] Coordenada Y del primer punto hacia el que se traza la línea desde el centro del rectángulo para obtener el límite del arco.

x4

[in] Coordenada X del segundo punto hacia el que se traza la línea desde el centro del rectángulo para obtener el límite del arco.

y4

[in] Coordenada Y del segundo punto hacia el que se traza la línea desde el centro del rectángulo para obtener el límite del arco.

clr

[in] Color en formato ARGB. Para transformar un color en el formato ARGB, use la función ColorToARGB().

© 2000-2025, MetaQuotes Ltd.

Dibuja el arco de la elipse con el centro en el punto (x,y), inscrito en un rectángulo con los radios rx y ry. Los límites del arco se cortan con los rayos desde el centro de la elipse, establecidos con los ángulos fi3 y fi4.

```
void  Arc(
   int          x,        // coordenada x del centro de la elipse
   int          y,        // coordenada Y del centro de la elipse
   int          rx,       // radio de la elipse según la coordenada X
   int          ry,       // radio de la elipse según la coordenada Y
   int          fi3,      // ángulo del rayo desde el centro de elipse, que establece el
   int          fi4,      // ángulo del rayo desde el centro de la elipse, que establece
   const uint   clr       // color
   );
```

Dibuja el arco de la elipse con el centro en el punto (x,y), inscrito en un rectángulo con los radios rx y ry, y retorna las coordenadas de los límites del arco. Los límites del arco se cortan con los rayos desde el centro de la elipse, establecidos con los ángulos fi3 y fi4.

```
void  Arc(
   int          x,        // coordenada x del centro de la elipse
   int          y,        // coordenada Y del centro de la elipse
   int          rx,       // radio de la elipse según la coordenada X
   int          ry,       // radio de la elipse según la coordenada Y
   int          fi3,      // ángulo del rayo desde el centro de elipse, que establece el
   int          fi4,      // ángulo del rayo desde el centro de la elipse, que establece
   int&         x3,       // coordenada X del primer límite del arco
   int&         y3,       // coordenada Y del primer límite del arco
   int&         x4,       // coordenada X del segundo límite del arco
   int&         y4,       // coordenada Y del segundo límite del arco
   const uint   clr       // color
   );
```

Parámetros

x

[in] Coordenada X del centro de la elipse.

y

[in] Coordenada Y del centro de la elipse

rx

[in] Radio de la elipse según la coordenada X, en píxeles.

ry

[in] Radio de la elipse según la coordenada Y, en píxeles.

fi3

[in] Ángulo en radianes que establece el primer límite del arco

fi4

© 2000-2025, MetaQuotes Ltd.

[in] Ángulo en radianes que establece el segundo límite del arco

x3

[out] Variable para obtener la coordenada X del primer límite del arco.

y3

[out] Variable para obtener la coordenada Y del primer límite del arco.

x4

[out] Variable para obtener la coordenada X del segundo límite del arco.

y4

[out] Variable para obtener la coordenada Y del segundo límite del arco.

clr

[in] Color en formato ARGB. Para transformar un color en el formato ARGB, use la función ColorToARGB().

Ejemplos de llamada de los métodos de clase:

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- create canvas
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Error creating canvas: ",GetLastError());
     }
//--- clear canvas
   canvas.Erase(clrWhite);
//--- draw rectangle
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- draw first arc
   canvas.Arc(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrRed));
   int x1,y1,x2,y2;
//--- draw second arc
   canvas.Arc(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,x1,y1,x2,y2,ColorToARGB(cl
//--- print coordinates of arc
   PrintFormat("First point of arc at (%G,%G), second point of arc at (%G,%G)",x1,y1,x
   canvas.CircleAA(x1,y1,3, ColorToARGB(clrRed));
   canvas.CircleAA(x2,y2,3, ColorToARGB(clrBlue));
//--- show updated canvas
   canvas.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Pie

Dibuja un sector coloreado de la elipse, inscrito en el rectángulo con los ángulos (x1,y1) y (x2,y2). Los límites del sector se cortan con las líneas desde el centro de la elipse, que buscan los dos puntos con las coordenadas (x3,y3) y (x4,y4).

```
void  Pie(
   int          x1,      // coordenada X del ángulo superior izquierdo del rectángulo
   int          y1,      // coordenada Y del ángulo superior izquierdo del rectángulo
   int          x2,      // coordenada X del ángulo inferior derecho del rectángulo
   int          y2,      // coordenada Y del ángulo inferior derecho del rectángulo
   int          x3,      // coordenada X del primer punto para encontrar el límite del
   int          y3,      // coordenada Y del primer punto para encontrar el límite del
   int          x4,      // coordenada X del segundo punto para encontrar el límite del
   int          y4,      // coordenada Y del segundo punto para encontrar el límite del
   const uint   clr,     // color de la línea
   const uint   fill_clr // color del rellenado
   );
```

Parámetros

x1

[in] Coordenada X del ángulo superior izquierdo que define el rectángulo.

y1

[in] Coordenada Y del ángulo superior izquierdo que define el rectángulo.

x2

[in] Coordenada X del ángulo inferior derecho que define el rectángulo.

y2

[in] Coordenada Y del ángulo inferior derecho que define el rectángulo.

x3

[in] Coordenada X del primer punto hacia el que se traza la línea desde el centro del rectángulo para obtener el límite del arco.

y3

[in] Coordenada Y del primer punto hacia el que se traza la línea desde el centro del rectángulo para obtener el límite del arco.

x4

[in] Coordenada X del segundo punto hacia el que se traza la línea desde el centro del rectángulo para obtener el límite del arco.

y4

[in] Coordenada Y del segundo punto hacia el que se traza la línea desde el centro del rectángulo para obtener el límite del arco.

clr

[in] Color del límite del sector en formato ARGB.

fill_clr

© 2000-2025, MetaQuotes Ltd.

[in] Color del rellenado del sector en formato ARGB. Para transformar un color en el formato ARGB, use la función ColorToARGB().

Dibuja el sector coloreado de la elipse con el centro en el punto (x,y), inscrito en el rectángulo con los radios rx y ry. Los límites del sector se cortan con los rayos desde centro de la elipse, establecidos con los ángulos fi3 y fi4.

```
void  Pie(
   int          x,        // coordenada x del centro de la elipse
   int          y,        // coordenada Y del centro de la elipse
   int          rx,       // radio de la elipse según la coordenada X
   int          ry,       // radio de la elipse según la coordenada Y
   int          fi3,      // ángulo del rayo desde el centro de elipse, que establece el
   int          fi4,      // ángulo del rayo desde el centro de la elipse, que establece
   const uint   clr,      // color de la línea
   const uint   fill_clr // color del rellenado
   );
```

Dibuja el sector coloreado de la elipse con el centro en el punto (x,y), inscrito en un rectángulo con los radios rx y ry, y retorna las coordenadas de los límites del arco. Los límites del sector se cortan con los rayos desde centro de la elipse, establecidos con los ángulos fi3 y fi4.

```
void  Pie(
   int          x,        // coordenada x del centro de la elipse
   int          y,        // coordenada Y del centro de la elipse
   int          rx,       // radio de la elipse según la coordenada X
   int          ry,       // radio de la elipse según la coordenada Y
   int          fi3,      // ángulo del rayo desde el centro de elipse, que establece el
   int          fi4,      // ángulo del rayo desde el centro de la elipse, que establece
   int&         x3,       // coordenada X del primer límite del arco
   int&         y3,       // coordenada Y del primer límite del arco
   int&         x4,       // coordenada X del segundo límite del arco
   int&         y4,       // coordenada Y del segundo límite del arco
   const uint   clr,      // color de la línea
   const uint   fill_clr // color del rellenado
   );
```

Parámetros

x

[in] Coordenada X del centro de la elipse.

y

[in] Coordenada Y del centro de la elipse

rx

[in] Radio de la elipse según la coordenada X, en píxeles.

ry

[in] Radio de la elipse según la coordenada X, en píxeles.

fi3

© 2000-2025, MetaQuotes Ltd.

[in] Ángulo en radianes que establece el primer límite del arco

fi4

[in] Ángulo en radianes que establece el segundo límite del arco

x3

[out] Variable para obtener la coordenada X del primer límite del arco.

y3

[out] Variable para obtener la coordenada Y del primer límite del arco.

x4

[out] Variable para obtener la coordenada X del segundo límite del arco.

y4

[out] Variable para obtener la coordenada Y del segundo límite del arco.

clr

[in] Color del límite del sector en formato ARGB.

fill_clr

[in] Color del rellenado del sector en formato ARGB. Para transformar un color en el formato ARGB, use la función ColorToARGB().

Ejemplos de llamada de los métodos de clase:

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- create canvas
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Error creating canvas: ",GetLastError());
     }
//--- clear canvas
   canvas.Erase(clrWhite);
//--- draw rectangle
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- draw first pie
   canvas.Pie(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrBlue),ColorToARGB(c
//--- draw second pie
   canvas.Pie(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,ColorToARGB(clrGreen),Colo
//--- show updated canvas
   canvas.Update();
   DebugBreak();
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

FillPolygon

Dibuja un rectángulo coloreado.

```
void  FillPolygon(
   int&          x,      // matriz con las coordenadas de los puntos X del polígono
   int&          y,      // matriz con las coordenadas de los puntos Y del polígono
   const uint    clr     // color
   );
```

Parámetros

x

[in] Matriz que contiene las coordenadas de los puntos X del polígono.

y

[in] Matriz que contiene las coordenadas de los puntos Y del polígono.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

FillEllipse

Dibuja una elipse coloreada, inscrita en un rectángulo con las coordenadas establecidas.

```
void  FillPolygon(
   int          x1,      // coordenada X del ángulo superior izquierdo del rectángulo
   int          y1,      // coordenada Y del ángulo superior izquierdo del rectángulo
   int          x2,      // coordenada X del ángulo inferior derecho del rectángulo
   int          y2,      // coordenada Y del ángulo inferior derecho del rectángulo
   const uint   clr       // color de la elipse
   );
```

Parámetros

x1

[in] Coordenada X del ángulo superior izquierdo que define el rectángulo.

y1

[in] Coordenada Y del ángulo superior izquierdo que define el rectángulo.

x2

[in] Coordenada X del ángulo inferior derecho que define el rectángulo.

y2

[in] Coordenada Y del ángulo inferior derecho que define el rectángulo.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

GetDefaultColor

Retorna un color predeterminado según su índice.

```
static uint  GetDefaultColor(
   const uint  i      // índice
   );
```

Parámetros

i

[in] Índice para obtener el color.

Valor devuelto

Color.

© 2000-2025, MetaQuotes Ltd.

ChartObjectName

Devuelve el nombre del objeto gráfico enlazado.

```
string  ChartObjectName();
```

Valor devuelto

el nombre del objeto gráfico enlazado

© 2000-2025, MetaQuotes Ltd.

Circle

Dibuja un círculo

```
void  Circle(
   int          x,        // coordenada X
   int          y,        // coordenada Y
   int          r,        // radio
   const uint   clr       // color
   );
```

Parámetros

x

[in] Coordenada X del centro del círculo.

y

[in] Coordenada Y del centro del círculo

r

[in] Radio del círculo.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

CircleAA

Dibuja un círculo utilizando un algoritmo antialiasing

```
void  CircleAA(
   const int      x,        // coordenada X
   const int      y,        // coordenada Y
   const double   r,        // radio
   const uint     clr       // color
   );
```

Parámetros

x

[in] Coordenada X del centro del círculo.

y

[in] Coordenada Y del centro del círculo

r

[in] Radio del círculo.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

CircleWu

Dibuja un círculo utilizando un algoritmo antialiasing Wu.

```
void  CircleWu(
   const int     x,        // coordenada X
   const int     y,        // coordenada Y
   const double  r,        // radio
   const uint    clr       // color
   );
```

Parámetros

x

[in] Coordenada X del centro del círculo.

y

[in] Coordenada Y del centro del círculo

r

[in] Radio del círculo.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

Create

Crea un recurso gráfico sin enlazarlo a un objeto gráfico.

```
virtual bool  Create(
   const string       name,                              // nombre
   const int          width,                             // anchura
   const int          height,                            // altura
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // formato
   );
```

Parámetros

name

[in] Nombre base de un recurso gráfico. En el momento de su creación, se crea un nombre de recurso añadiendo una cadena pseudoaleatoria.

width

[in] Anchura (a lo largo del eje X) en píxeles.

height

[in] Altura (a lo largo del eje Y) en píxeles.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Método de procesamiento del color. Consultar la descripción de la función ResourceCreate() para más información acerca de los métodos de procesamiento del color.

Valor devuelto

true - si se ejecuta correctamente, en caso contrario - false

© 2000-2025, MetaQuotes Ltd.

CreateBitmap

Crea un recurso gráfico enlazado a un objeto gráfico.

1. Crea un recurso gráfico en la ventana principal del gráfico actual.

```
bool  CreateBitmap(
   const string       name,                                  // nombre
   const datetime     time,                                  // hora
   const double       price,                                 // precio
   const int          width,                                 // anchura
   const int          height,                                // altura
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // formato
   );
```

2. Crea un recurso gráfico utilizando un ID de gráfico y un número de subventana.

```
bool  CreateBitmap(
   const long         chart_id,                              // ID de gráfico
   const int          subwin,                                // número de subventana
   const string       name,                                  // nombre
   const datetime     time,                                  // hora
   const double       price,                                 // precio
   const int          width,                                 // anchura
   const int          height,                                // altura
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // formato
   );
```

Parámetros

chart_id

[in] ID de gráfico para crear un objeto.

subwin

[in] Número de subventana de gráfico para crear un objeto.

name

[in] Nombre del objeto gráfico y nombre base del recurso gráfico.

time

[in] Coordenada temporal del punto de anclaje del objeto gráfico.

price

[in] Coordenada del precio del punto de anclaje del objeto gráfico.

width

[in] Anchura del recurso gráfico (a lo largo del eje X) en píxeles.

height

[in] Altura del recurso gráfico (a lo largo del eje Y) en píxeles.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2025, MetaQuotes Ltd.

[in] Método de procesamiento del color. Consultar la descripción de la función ResourceCreate() para más información acerca de los métodos de procesamiento del color.

Valor devuelto

true - si se ejecuta correctamente, en caso contrario - false

Nota

Si se utiliza la primera versión de la función, el objeto se crea en la ventana principal del gráfico actual.

El tamaño del objeto coincide con el tamaño del recurso gráfico.

© 2000-2025, MetaQuotes Ltd.

CreateBitmapLabel

Crea un recurso gráfico enlazado a un objeto gráfico.

1. Crea un recurso gráfico en la ventana principal del gráfico actual.

```
bool  CreateBitmapLabel(
   const string       name,                              // nombre
   const int          x,                                 // coordenada X
   const int          y,                                 // coordenada Y
   const int          width,                             // anchura
   const int          height,                            // altura
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // formato
   );
```

2. Crea un recurso gráfico utilizando un ID de gráfico y un número de subventana.

```
bool  CreateBitmapLabel(
   const long         chart_id,                          // ID de gráfico
   const int          subwin,                            // número de subventana
   const string       name,                              // nombre
   const int          x,                                 // coordenada X
   const int          y,                                 // coordenada Y
   const int          width,                             // anchura
   const int          height,                            // altura
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // formato
   );
```

Parámetros

chart_id

[in] ID de gráfico para crear un objeto.

subwin

[in] Número de subventana de gráfico para crear un objeto.

name

[in] Nombre del objeto gráfico y nombre base del recurso gráfico.

x

[in] Coordenada X del punto de anclaje del objeto gráfico.

y

[in] Coordenada Y del punto de anclaje del objeto gráfico.

width

[in] Anchura del recurso gráfico (a lo largo del eje X) en píxeles.

height

[in] Altura del recurso gráfico (a lo largo del eje Y) en píxeles.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2025, MetaQuotes Ltd.

[in] Método de procesamiento del color. Consultar la descripción de la función ResourceCreate() para más información acerca de los métodos de procesamiento del color.

Valor devuelto

true - si se ejecuta correctamente, en caso contrario - false

Nota

Si se utiliza la primera versión de la función, el objeto se crea en la ventana principal del gráfico actual.

El tamaño del objeto coincide con el tamaño del recurso gráfico.

© 2000-2025, MetaQuotes Ltd.

Destroy

Destruye un objeto gráfico.

```
void  Destroy();
```

Nota

Si un recurso gráfico ha sido enlazado a un objeto gráfico, este último se elimina.

© 2000-2025, MetaQuotes Ltd.

Ellipse

Dibuja una elipse por medio de dos puntos.

```
void  Ellipse(
   int          x1,      // coordenada X
   int          y1,      // coordenada Y
   int          x2,      // coordenada X
   int          y2,      // coordenada Y
   const uint   clr      // color
   );
```

Parámetros

x1

[in] Coordenada X del primer punto que determina la elipse.

y1

[in] Coordenada Y del primer punto que determina la elipse.

x2

[in] Coordenada X del segundo punto que determina la elipse.

y2

[in] Coordenada Y del segundo punto que determina la elipse.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

EllipseAA

Dibuja una elipse por medio de dos puntos y un algoritmo antialiasing.

```
void  EllipseAA(
   int          x1,      // coordenada X
   int          y1,      // coordenada Y
   int          x2,      // coordenada X
   int          y2,      // coordenada Y
   const uint   clr      // color
   );
```

Parámetros

x1

[in] Coordenada X del primer punto que determina la elipse.

y1

[in] Coordenada Y del primer punto que determina la elipse.

x2

[in] Coordenada X del segundo punto que determina la elipse.

y2

[in] Coordenada Y del segundo punto que determina la elipse.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

EllipseWu

Dibuja una elipse por medio de dos puntos y un algoritmo antialiasing Wu.

```
void  Ellipse(
   int          x1,      // coordenada X
   int          y1,      // coordenada Y
   int          x2,      // coordenada X
   int          y2,      // coordenada Y
   const uint   clr      // color
   );
```

Parámetros

x1

[in] Coordenada X del primer punto que determina la elipse.

y1

[in] Coordenada Y del primer punto que determina la elipse.

x2

[in] Coordenada X del segundo punto que determina la elipse.

y2

[in] Coordenada Y del segundo punto que determina la elipse.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

Erase

Borra o rellena con el color especificado.

```
void  Erase(
   const uint  clr=0      // color
   );
```

Parámetros

clr=0

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

Fill

Rellena un área.

```
void  Fill(
   int          x,        // coordenada X
   int          y,        // coordenada Y
   const uint   clr       // color
   );
```

Parámetros

x

[in] Coordenada X correspondiente al punto de inicio del relleno.

y

[in] Coordenada Y correspondiente al punto de inicio del relleno.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

FillCircle

Dibuja un círculo con relleno.

```
void  FillCircle(
   int          x,        // coordenada X
   int          y,        // coordenada Y
   int          r,        // radio
   const uint   clr       // color
   );
```

Parámetros

x

[in] Coordenada X del centro del círculo con relleno.

y

[in] Coordenada Y del centro del círculo con relleno.

r

[in] Radio del círculo con relleno.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

FillRectangle

Dibuja un rectángulo con relleno.

```
void  FillRectangle(
   int          x1,      // coordenada X
   int          y1,      // coordenada Y
   int          x2,      // coordenada X
   int          y2,      // coordenada Y
   const uint   clr      // color
   );
```

Parámetros

x1

[in] Coordenada X del primer punto que determina el rectángulo.

y1

[in] Coordenada Y del primer punto que determina el rectángulo.

x2

[in] Coordenada X del segundo punto que determina el rectángulo.

y2

[in] Coordenada Y del segundo punto que determina el rectángulo.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

FillTriangle

Dibuja un triángulo con relleno.

```
void  FillTriangle(
   int          x1,       // coordenada X
   int          y1,       // coordenada Y
   int          x2,       // coordenada X
   int          y2,       // coordenada Y
   int          x3,       // coordenada X
   int          y3,       // coordenada Y
   const uint   clr       // color
   );
```

Parámetros

x1

[in] Coordenada X del primer vértice del triángulo.

y1

[in] Coordenada Y del primer vértice del triángulo.

x2

[in] Coordenada X del segundo vértice del triángulo.

y2

[in] Coordenada Y del segundo vértice del triángulo.

x3

[in] Coordenada X del tercer vértice del triángulo.

y3

[in] Coordenada Y del tercer vértice del triángulo.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

FontAngleGet

Devuelve el ángulo de inclinación de la fuente.

```
uint  FontAngleGet();
```

Valor devuelto

ángulo de inclinación de la fuente

© 2000-2025, MetaQuotes Ltd.

FontAngleSet

Establece el ángulo de inclinación de la fuente.

```
bool  FontAngleSet(
   uint  angle      // ángulo
   );
```

Parámetros

angle

[in] Ángulo de inclinación de la fuente en décimas de grado.

Valor devuelto

true - si se ejecuta correctamente, en caso contrario - false

© 2000-2025, MetaQuotes Ltd.

FontFlagsGet

Devuelve las banderas de la fuente.

```
uint  FontFlagsGet();
```

Valor devuelto

banderas de la fuente

© 2000-2025, MetaQuotes Ltd.

FontFlagsSet

Establece las banderas de la fuente.

```
bool  FontFlagsSet(
   uint  flags      // banderas
   );
```

Parámetros

flags

[in] Banderas de creación de la fuente. Consultar la descripción de la función TextSetFont() para más información sobre las banderas.

Valor devuelto

true - si se ejecuta correctamente, en caso contrario - false

© 2000-2025, MetaQuotes Ltd.

FontGet

Devuelve los parámetros de la fuente actual.

```
void  FontGet(
   string&  name,      // nombre
   int&     size,      // tamaño
   uint&    flags,     // banderas
   uint&    angle      // ángulo de inclinación
   );
```

Parámetros

name

[out] Referencia a la variable que almacena el nombre de la fuente.

size

[out] Referencia a la variable que almacena el tamaño de la fuente.

flags

[out] Referencia a la variable que almacena las banderas de la fuente.

angle

[out] Referencia a la variable que almacena el ángulo de inclinación de la fuente.

© 2000-2025, MetaQuotes Ltd.

FontNameGet

Devuelve el nombre de la fuente.

```
string  FontNameGet();
```

Valor devuelto

nombre de la fuente

© 2000-2025, MetaQuotes Ltd.

FontNameSet

Establece el nombre de la fuente.

```
bool  FontNameSet(
   string  name      // nombre
   );
```

Parámetros

name

[in] Nombre de la fuente. Por ejemplo, "Arial".

Valor devuelto

true - si se ejecuta correctamente, en caso contrario - false

© 2000-2025, MetaQuotes Ltd.

FontSet

Establece la fuente actual.

```
bool  FontSet(
   const string  name,         // nombre
   const int     size,         // tamaño
   const uint    flags=0,      // banderas
   const uint    angle=0       // ángulo
   );
```

Parámetros

name

[in] Nombre de la fuente. Por ejemplo, "Arial".

size

[in] Tamaño de la fuente. Consultar la descripción de la función TextSetFont() para más información sobre cómo establecer el tamaño.

flags=0

[in] Banderas de creación de la fuente. Consultar la descripción de la función TextSetFont() para más información sobre las banderas.

angle=0

[in] Ángulo de inclinación de la fuente en décimas de grado.

Valor devuelto

true - si se ejecuta correctamente, en caso contrario - false

© 2000-2025, MetaQuotes Ltd.

FontSizeGet

Devuelve el tamaño de la fuente.

```
int  FontSizeGet();
```

Valor devuelto

tamaño de la fuente

© 2000-2025, MetaQuotes Ltd.

FontSizeSet

Establece el tamaño de la fuente.

```
bool  FontSizeSet(
   int  size      // tamaño
   );
```

Parámetros

size

[in] Tamaño de la fuente. Consultar la descripción de la función TextSetFont() para más información sobre cómo establecer el tamaño.

Valor devuelto

true - si se ejecuta correctamente, en caso contrario - false

© 2000-2025, MetaQuotes Ltd.

Height

Devuelve la altura de un recurso gráfico.

```
int  Height();
```

Valor devuelto

altura de un recurso gráfico

© 2000-2025, MetaQuotes Ltd.

Line

Dibuja un segmento de línea a mano alzada.

```
void  Line(
   int          x1,       // coordenada X
   int          y1,       // coordenada Y
   int          x2,       // coordenada X
   int          y2,       // coordenada Y
   const uint   clr       // color
   );
```

Parámetros

x1

[in] Coordenada X del primer punto del segmento.

y1

[in] Coordenada Y del primer punto del segmento.

x2

[in] Coordenada X del segundo punto del segmento.

y2

[in] Coordenada Y del segundo punto del segmento.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

LineAA

Dibuja un segmento de línea a mano alzada utilizando un algoritmo antialiasing.

```
void  LineAA(
   const int   x1,                  // coordenada X
   const int   y1,                  // coordenada Y
   const int   x2,                  // coordenada X
   const int   y2,                  // coordenada Y
   const uint  clr,                 // color
   const uint  style=UINT_MAX       // estilo de línea
   );
```

Parámetros

x1

[in] Coordenada X del primer punto del segmento.

y1

[in] Coordenada Y del primer punto del segmento.

x2

[in] Coordenada X del segundo punto del segmento.

y2

[in] Coordenada Y del segundo punto del segmento.

clr

[in] Color en formato ARGB.

style=UINT_MAX

[in] El estilo de línea se corresponde con alguno de los valores de ENUM_LINE_STYLE, o bien con un valor personalizado.

© 2000-2025, MetaQuotes Ltd.

LineWu

Dibuja un segmento de línea a mano alzada utilizando un algoritmo antialiasing Wu.

```
void  LineWu(
   const int   x1,                  // coordenada X
   const int   y1,                  // coordenada Y
   const int   x2,                  // coordenada X
   const int   y2,                  // coordenada Y
   const uint  clr,                 // color
   const uint  style=UINT_MAX       // estilo de línea
   );
```

Parámetros

x1

[in] Coordenada X del primer punto del segmento.

y1

[in] Coordenada Y del primer punto del segmento.

x2

[in] Coordenada X del segundo punto del segmento.

y2

[in] Coordenada Y del segundo punto del segmento.

clr

[in] Color en formato ARGB.

style=UINT_MAX

[in] El estilo de línea se corresponde con alguno de los valores de ENUM_LINE_STYLE, o bien con un valor personalizado.

© 2000-2025, MetaQuotes Ltd.

LineHorizontal

Dibuja un segmento de línea horizontal.

```
void  LineHorizontal(
   int          x1,       // coordenada X
   int          x2,       // coordenada X
   int          y,        // coordenada Y
   const uint   clr       // color
   );
```

Parámetros

x1

[in] Coordenada X del primer punto del segmento.

x2

[in] Coordenada X del segundo punto del segmento.

y

[in] Coordenada Y del segmento.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

LineVertical

Dibuja un segmento de línea vertical.

```
void  LineVertical(
   int          x,        // coordenada X
   int          y1,       // coordenada Y
   int          y2,       // coordenada Y
   const uint   clr       // color
   );
```

Parámetros

x

[in] Coordenada X del segmento.

y1

[in] Coordenada Y del primer punto del segmento.

y2

[in] Coordenada Y del segundo punto del segmento.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

LineStyleSet

Establece el estilo de línea.

```
void  LineStyleSet(
   const uint  style      // estilo
   );
```

Parámetros

style

[in] Estilo de línea.

Nota

El parámetro de entrada se corresponde con cualquiera de los valores de la enumeración ENUM_LINE_STYLE. También se puede crear un estilo de línea personalizado.

© 2000-2025, MetaQuotes Ltd.

LineThick

Dibuja un segmento de una línea aleatoria con un grosor establecido usando un algoritmo de suavizado.

```
void  LineThick(
   const int       x1,            // coordenada X del primer punto del segmento
   const int       y1,            // coordenada Y del primer punto del segmento
   const int       x2,            // coordenada X del segundo punto del segmento
   const int       y2,            // coordenada Y del segundo punto del segmento
   const uint      clr,           // color
   const int       size,          // grosor de la línea
   const uint      style,         // estilo de la línea
   ENUM_LINE_END   end_style      // estilo de los extremos de la línea
   )
```

Parámetros

x1

[in] Coordenada X del primer punto del segmento.

y1

[in] Coordenada Y del primer punto del segmento.

x2

[in] Coordenada X del segundo punto del segmento.

y2

[in] Coordenada Y del segundo punto del segmento.

clr

[in] Color en formato ARGB.

size

[in] Grosor de la línea.

style

[in] El estilo de la línea es uno de los valores de la enumeración ENUM_LINE_STYLE o un valor personalizado.

end_style

[in] El estilo de los extremos de la línea es uno de los valores de la enumeración ENUM_LINE_END

ENUM_LINE_END

Identificador	Descripción
LINE_END_ROUND	Los extremos de las líneas se redondean.
LINE_END_BUTT	Los extremos de las líneas se cortan.

© 2000-2025, MetaQuotes Ltd.

Identificador	Descripción
LINE_END_SQUARE	La línea finaliza con un rectángulo coloreado.

© 2000-2025, MetaQuotes Ltd.

LineThickVertical

Dibuja un segmento vertical de una línea aleatoria con un grosor establecido usando un algoritmo de suavizado.

```
void  LineThickVertical(
   const int       x,              // coordenada X del segmento
   const int       y1,             // coordenada Y del primer punto del segmento
   const int       y2,             // coordenada Y del segundo punto del segmento
   const uint      clr,            // color
   const int       size,           // grosor de la línea
   const uint      style,          // estilo de la línea
   ENUM_LINE_END   end_style       // estilo de los extremos de la línea
   )
```

Parámetros

x

[in] Coordenada X del segmento.

y1

[in] Coordenada Y del primer punto del segmento.

y2

[in] Coordenada Y del segundo punto del segmento.

clr

[in] Color en formato ARGB.

size

[in] Grosor de la línea.

style

[in] El estilo de la línea es uno de los valores de la enumeración ENUM_LINE_STYLE o un valor personalizado.

end_style

[in] El estilo de los extremos de la línea es uno de los valores de la enumeración ENUM_LINE_END.

© 2000-2025, MetaQuotes Ltd.

LineThickHorizontal

Dibuja un segmento horizontal de una línea aleatoria con un grosor establecido con suavizado.

```
void  LineThickHorizontal(
   const int       x1,            // coordenada X del primer punto del segmento
   const int       x2,            // coordenada X del segundo punto del segmento
   const int       y,             // coordenada Y del segmento
   const uint      clr,           // color
   const int       size,          // grosor de la línea
   const uint      style,         // estilo de la línea
   ENUM_LINE_END   end_style      // estilo de los extremos de la línea
   )
```

Parámetros

x1

[in] Coordenada X del primer punto del segmento.

x2

[in] Coordenada X del segundo punto del segmento.

y

[in] Coordenada Y del segmento.

clr

[in] Color en formato ARGB.

size

[in] Grosor de la línea.

style

[in] El estilo de la línea es uno de los valores de la enumeración ENUM_LINE_STYLE o un valor personalizado.

end_style

[in] El estilo de los extremos de la línea es uno de los valores de la enumeración ENUM_LINE_END.

© 2000-2025, MetaQuotes Ltd.

LoadFromFile

Lee una imagen de un archivo BMP.

```
bool  LoadFromFile(
   const string  filename      // nombre del archivo
   );
```

Parámetros

filename

[in] Nombre del archivo (incluyendo la extensión "BMP").

Valor devuelto

true - si se ejecuta correctamente, en caso contrario - false

© 2000-2025, MetaQuotes Ltd.

PixelGet

Devuelve el color del punto correspondiente a las coordenadas especificadas.

```
uint  PixelGet(
   const int  x,      // coordenada X
   const int  y       // coordenada Y
   );
```

Parámetros

x

[in] Coordenada X del punto.

y

[in] Coordenada Y del punto.

Valor devuelto

Color del punto en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

PixelSet

Establece el color del punto correspondiente a las coordenadas especificadas.

```
void  PixelSet(
   const int   x,       // coordenada X
   const int   y,       // coordenada Y
   const uint  clr      // color
   );
```

Parámetros

x

[in] Coordenada X del punto.

y

[in] Coordenada Y del punto.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

PixelSetAA

Dibuja un punto utilizando un algoritmo antialiasing.

```
void  PixelSetAA(
   const double  x,        // coordenada X
   const double  y,        // coordenada Y
   const uint    clr       // color
   );
```

Parámetros

x

[in] Coordenada X del punto.

y

[in] Coordenada Y del punto.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

Polygon

Dibuja un polígono.

```
void  Polygon(
   int&         x[],     // array de coordenadas X
   int&         y[],     // array de coordenadas Y
   const uint   clr      // color
   );
```

Parámetros

x[]

[in] Array de coordenadas X de los puntos de un polígono.

y[]

[in] Array de coordenadas Y de los puntos de un polígono.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

PolygonAA

Dibuja un polígono utilizando un algoritmo antialiasing.

```
void  PolygonAA(
   int&         x[],                // array de coordenadas X
   int&         y[],                // array de coordenadas Y
   const uint   clr,                // color
   const uint   style=UINT_MAX      // estilo de línea
   );
```

Parámetros

x[]

[in] Array de coordenadas X de los puntos de un polígono.

y[]

[in] Array de coordenadas Y de los puntos de un polígono.

clr

[in] Color en formato ARGB.

style=UINT_MAX

[in] El estilo de línea se corresponde con alguno de los valores de ENUM_LINE_STYLE, o bien con un valor personalizado.

© 2000-2025, MetaQuotes Ltd.

PolygonWu

Dibuja un polígono utilizando un algoritmo antialiasing Wu.

```
void  PolygonWu(
   int&         x[],                // array de coordenadas X
   int&         y[],                // array de coordenadas Y
   const uint   clr,                // color
   const uint   style=UINT_MAX      // estilo de línea
   );
```

Parámetros

x[]

[in] Array de coordenadas X de los puntos de un polígono.

y[]

[in] Array de coordenadas Y de los puntos de un polígono.

clr

[in] Color en formato ARGB.

style=UINT_MAX

[in] El estilo de línea se corresponde con alguno de los valores de ENUM_LINE_STYLE, o bien con un valor personalizado.

© 2000-2025, MetaQuotes Ltd.

PolygonThick

Dibuja un polígono con el grosor establecido usando un algoritmo de suavizado.

```
void  PolygonThick(
   const int&      x[],          // matriz de coordenadas X de los puntos del polígono
   const int&      y[],          // matriz de coordenadas Y de los puntos del polígono
   const uint      clr,          // color
   const int       size,         // grosor de la línea
   const uint      style,        // estilo de la línea
   ENUM_LINE_END   end_style     // estilo de los extremos de la línea
   )
```

Parámetros

x[]

[in] Matriz de coordenadas X de los puntos del polígono.

y[]

[in] Matriz de coordenadas Y de los puntos del polígono.

clr

[in] Color en formato ARGB.

size

[in] Grosor de la línea.

style

[in] El estilo de la línea es uno de los valores de la enumeración ENUM_LINE_STYLE o un valor personalizado.

end_style

[in] El estilo de los extremos de la línea es uno de los valores de la enumeración ENUM_LINE_END.

© 2000-2025, MetaQuotes Ltd.

PolygonSmooth

Dibuja un polígono con el grosor establecido usando dos algoritmos de suavizado de forma secuencial. En primer lugar, usando como base las curvas de Bezier, se suavizan segmentos aparte. A continuación, para aumentar la calidad de dibujado se aplica un algoritmo bitmap al polígono construido a partir de estos segmentos.

```
void  PolygonSmooth(
   int&             x[],                        // matriz de coordenadas X de los pu
   int&             y[],                        // matriz de coordenadas Y de los pu
   const uint       clr,                        // color
   const int        size,                       // grosor de la línea
   ENUM_LINE_STYLE  style=STYLE_SOLID,          // estilo de la línea
   ENUM_LINE_END    end_style=LINE_END_ROUND,   // estilo de los extremos de la líne
   double           tension=0.5,                // valor del parámetro de suavizado
   double           step=10                     // longitud de las líneas que se apr
   )
```

Parámetros

&x[]

[in] Matriz de coordenadas X de los puntos del polígono.

&y[]

[in] Matriz de coordenadas Y de los puntos del polígono.

clr

[in] Color en formato ARGB.

size

[in] Grosor de la línea.

style=STYLE_SOLID

[in] El estilo de la línea es uno de los valores de la enumeración ENUM_LINE_STYLE o un valor personalizado.

end_style=LINE_END_ROUND

[in] El estilo de los extremos de la línea es uno de los valores de la enumeración ENUM_LINE_END.

tension=0.5

[in] Valor del parámetro de suavizado.

step=10

[in] Longitud de las líneas que se aproximan.

© 2000-2025, MetaQuotes Ltd.

Polyline

Dibuja una polilínea.

```
void  Polyline(
   int&         x[],     // array de coordenadas X
   int&         y[],     // array de coordenadas Y
   const uint   clr      // color
   );
```

Parámetros

x[]

[in] Array de coordenadas X de una polilínea.

y[]

[in] Array de coordenadas Y de una polilínea.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

PolylineSmooth

Dibuja una línea quebrada con el grosor establecido usando dos algoritmos de suavizado de forma secuencial. En primer lugar, usando como base las curvas de Bezier, se suavizan segmentos aparte de la línea. A continuación, para aumentar la calidad de dibujado se aplica un algoritmo bitmap a la línea quebrada construida a partir de estos segmentos.

```
void  PolylineSmooth(
   const int&       x[],                        // matriz de coordenadas X de los pur
   const int&       y[],                        // matriz de coordenadas Y de los pur
   const uint       clr,                        // color
   const int        size,                       // grosor de la línea
   ENUM_LINE_STYLE  style=STYLE_SOLID,          // estilo de la línea
   ENUM_LINE_END    end_style=LINE_END_ROUND,   // estilo de los extremos de la línea
   double           tension=0.5,                // valor del parámetro de suavizado
   double           step=10                     // salto de aproximación
   )
```

Parámetros

&x[]

[in] Matriz de coordenadas X de los puntos de la línea quebrada.

&y[]

[in] Matriz de coordenadas Y de los puntos de la línea quebrada.

clr

[in] Color en formato ARGB.

size

[in] Grosor de la línea.

style=STYLE_SOLID

[in] El estilo de la línea es uno de los valores de la enumeración ENUM_LINE_STYLE o un valor personalizado.

end_style=LINE_END_ROUND

[in] El estilo de los extremos de la línea es uno de los valores de la enumeración ENUM_LINE_END.

tension=0.5

[in] Valor del parámetro de suavizado.

step=10

[in] Salto de aproximación.

© 2000-2025, MetaQuotes Ltd.

PolylineThick

Dibuja una línea quebrada con el grosor establecido usando algoritmo de suavizado.

```
void  PolylineThick(
   const int       &x[],          // matriz de coordenadas X de los puntos de la línea
   const int       &y[],          // matriz de coordenadas Y de los puntos de la línea
   const uint      clr,           // color
   const int       size,          // grosor de la línea
   const uint      style,         // estilo de la línea
   ENUM_LINE_END   end_style      // estilo de los extremos de la línea
   )
```

Parámetros

&x[]

[in] Matriz de coordenadas X de los puntos de la línea quebrada.

&y[]

[in] Matriz de coordenadas Y de los puntos de la línea quebrada.

clr

[in] Color en formato ARGB.

size

[in] Grosor de la línea.

style

[in] El estilo de la línea es uno de los valores de la enumeración ENUM_LINE_STYLE o un valor personalizado.

end_style

[in] El estilo de los extremos de la línea es uno de los valores de la enumeración ENUM_LINE_END

© 2000-2025, MetaQuotes Ltd.

PolylineWu

Dibuja una polilínea utilizando un algoritmo antialiasing Wu.

```
void  PolylineWu(
   int&         x[],                // array de coordenadas X
   int&         y[],                // array de coordenadas Y
   const uint   clr,                // color
   const uint   style=UINT_MAX      // estilo de línea
   );
```

Parámetros

x[]

[in] Array de coordenadas X de una polilínea.

y[]

[in] Array de coordenadas Y de una polilínea.

clr

[in] Color en formato ARGB.

style=UINT_MAX

[in] El estilo de línea se corresponde con alguno de los valores de ENUM_LINE_STYLE, o bien con un valor personalizado.

© 2000-2025, MetaQuotes Ltd.

PolylineAA

Dibuja una polilínea utilizando un algoritmo antialiasing.

```
void  PolylineAA(
   int&         x[],                // array de coordenadas X
   int&         y[],                // array de coordenadas Y
   const uint   clr,                // color
   const uint   style=UINT_MAX      // estilo de línea
   );
```

Parámetros

x[]

[in] Array de coordenadas X de una polilínea.

y[]

[in] Array de coordenadas Y de una polilínea.

clr

[in] Color en formato ARGB.

style=UINT_MAX

[in] El estilo de línea se corresponde con alguno de los valores de ENUM_LINE_STYLE, o bien con un valor personalizado.

© 2000-2025, MetaQuotes Ltd.

Rectangle

Dibuja un rectángulo por medio de dos puntos.

```
void  Rectangle(
   int          x1,      // coordenada X
   int          y1,      // coordenada Y
   int          x2,      // coordenada X
   int          y2,      // coordenada Y
   const uint   clr      // color
   );
```

Parámetros

x1

[in] Coordenada X del primer punto que determina el rectángulo.

y1

[in] Coordenada Y del primer punto que determina el rectángulo.

x2

[in] Coordenada X del segundo punto que determina el rectángulo.

y2

[in] Coordenada Y del segundo punto que determina el rectángulo.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

Resize

Cambia el tamaño de un recurso gráfico.

```
bool  Resize(
   const int  width,       // anchura
   const int  height       // altura
   );
```

Parámetros

width

[in] Nueva anchura del recurso gráfico.

height

[in] Nueva altura del recurso gráfico.

Valor devuelto

true - si se ejecuta correctamente, en caso contrario - false

Nota

La imagen anterior no queda guardada cuando se cambia el tamaño.

© 2000-2025, MetaQuotes Ltd.

ResourceName

Devuelve el nombre del recurso gráfico.

```
string  ResourceName();
```

Valor devuelto

nombre del recurso gráfico

© 2000-2025, MetaQuotes Ltd.

TextHeight

Devuelve la altura del texto.

```
int  TextHeight(
   const string  text      // texto
   );
```

Parámetros

text

[in] Texto a medir.

Valor devuelto

altura del texto en píxeles

Nota

Para medir el texto se utiliza la fuente actual.

© 2000-2025, MetaQuotes Ltd.

TextOut

Muestra el texto.

```
void  TextOut(
   int          x,               // coordenada X
   int          y,               // coordenada Y
   string       text,            // texto
   const uint   clr,             // color
   uint         alignment=0      // alineación
   );
```

Parámetros

x

[in] Coordenada X del texto ancla.

y

[in] Coordenada Y del texto ancla.

text

[in] Texto a mostrar.

clr

[in] Color en formato ARGB.

alignment=0

[in] Método de anclaje del texto. Consultar la descripción de la función TextOut() para más información sobre los métodos de anclaje.

Nota

La fuente actual se utiliza para mostrar el texto.

© 2000-2025, MetaQuotes Ltd.

TextSize

Devuelve el tamaño del texto.

```
void  TextSize(
   const string  text,        // texto
   int&          width,       // anchura
   int&          height       // altura
   );
```

Parámetros

text

[in] Texto a medir.

width

[out] Referencia a la variable que almacena la anchura del texto.

height

[out] Referencia a la variable que almacena la altura del texto.

Nota

Para medir el texto se utiliza la fuente actual.

© 2000-2025, MetaQuotes Ltd.

TextWidth

Devuelve la anchura del texto.

```
int  TextWidth(
   const string  text      // texto
   );
```

Parámetros

text

[in] Texto a medir.

Valor devuelto

altura del texto en píxeles

Nota

Para medir el texto se utiliza la fuente actual.

© 2000-2025, MetaQuotes Ltd.

TransparentLevelSet

Establece el nivel de transparencia.

```
void  TransparentLevelSet(
   const uchar  value      // valor
   );
```

Parámetros

value

[in] Valor nuevo del nivel de transparencia.

Nota

0 significa transparencia completa, y 255 quiere decir opacidad completa.

Al establecer un nivel de transparencia se ve afectado todo lo que se ha dibujado previamente. El nivel de transparencia especificado no afecta a las construcciones futuras.

© 2000-2025, MetaQuotes Ltd.

Triangle

Dibuja un triángulo.

```
void  Triangle(
   int          x1,      // coordenada X
   int          y1,      // coordenada Y
   int          x2,      // coordenada X
   int          y2,      // coordenada Y
   int          x3,      // coordenada X
   int          y3,      // coordenada Y
   const uint   clr      // color
   );
```

Parámetros

x1

[in] Coordenada X del primer vértice del triángulo.

y1

[in] Coordenada Y del primer vértice del triángulo.

x2

[in] Coordenada X del segundo vértice del triángulo.

y2

[in] Coordenada Y del segundo vértice del triángulo.

x3

[in] Coordenada X del tercer vértice del triángulo.

y3

[in] Coordenada Y del tercer vértice del triángulo.

clr

[in] Color en formato ARGB.

© 2000-2025, MetaQuotes Ltd.

TriangleAA

Dibuja un triángulo utilizando un algoritmo antialiasing.

```
void  TriangleAA(
   const int   x1,                  // coordenada X
   const int   y1,                  // coordenada Y
   const int   x2,                  // coordenada X
   const int   y2,                  // coordenada Y
   const int   x3,                  // coordenada X
   const int   y3,                  // coordenada Y
   const uint  clr,                 // color
   const uint  style=UINT_MAX       // estilo de línea
   );
```

Parámetros

x1

[in] Coordenada X del primer vértice del triángulo.

y1

[in] Coordenada Y del primer vértice del triángulo.

x2

[in] Coordenada X del segundo vértice del triángulo.

y2

[in] Coordenada Y del segundo vértice del triángulo.

x3

[in] Coordenada X del tercer vértice del triángulo.

y3

[in] Coordenada Y del tercer vértice del triángulo.

clr

[in] Color en formato ARGB.

style=UINT_MAX

[in] El estilo de línea se corresponde con alguno de los valores de ENUM_LINE_STYLE, o bien con un valor personalizado.

© 2000-2025, MetaQuotes Ltd.

TriangleWu

Dibuja un triángulo utilizando un algoritmo antialiasing Wu.

```
void  TriangleWu(
   const int   x1,                  // coordenada X
   const int   y1,                  // coordenada Y
   const int   x2,                  // coordenada X
   const int   y2,                  // coordenada Y
   const int   x3,                  // coordenada X
   const int   y3,                  // coordenada Y
   const uint  clr,                 // color
   const uint  style=UINT_MAX       // estilo de línea
   );
```

Parámetros

x1

[in] Coordenada X del primer vértice del triángulo.

y1

[in] Coordenada Y del primer vértice del triángulo.

x2

[in] Coordenada X del segundo vértice del triángulo.

y2

[in] Coordenada Y del segundo vértice del triángulo.

x3

[in] Coordenada X del tercer vértice del triángulo.

y3

[in] Coordenada Y del tercer vértice del triángulo.

clr

[in] Color en formato ARGB.

style=UINT_MAX

[in] El estilo de línea se corresponde con alguno de los valores de ENUM_LINE_STYLE, o bien con un valor personalizado.

© 2000-2025, MetaQuotes Ltd.

Update

Muestra los cambios en la pantalla.

```
void  Update(
   const bool  redraw=true      // bandera
   );
```

Parámetros

redraw=true

Bandera del nuevo gráfico redibujado.

© 2000-2025, MetaQuotes Ltd.

Width

Devuelve la anchura de un recurso gráfico.

```
int  Width();
```

Valor devuelto

anchura del recurso gráfico

© 2000-2025, MetaQuotes Ltd.

CChartCanvas

Clase básica para implementar las clases que sirven para dibujar los gráficos y sus elementos.

Descripción

Esta clase incluye los métodos para trabajar con los elementos principales de cualquier gráfico: las coordenadas de los ejes y su trazado, la leyenda del gráfico, la cuadrícula, el fondo, etcétera. Aquí se ajustan los parámetros de representación de los elementos: visibilidad/invisibilidad, el color del texto, etcétera.

Declaración

```
class CChartCanvas : public CCanvas
```

Encabezamiento

```
#include <Canvas\Charts\ChartCanvas.mqh>
```

Jerarquía de herencia

CCanvas

CChartCanvas

Descendientes directos

CHistogramChart, CLineChart, CPieChart

Métodos de clase

Método	Acción
ColorBackground	Retorna y establece el color del fondo.
ColorBorder	Retorna y establece el color de los límites.
ColorText	Retorna y establece el color del texto.
ColorGrid	Retorna y establece el color de la cuadrícula.
MaxData	Retorna y establece el número máximo de datos permitido (series).
MaxDescrLen	Retorna y establece la longitud máxima de los descriptores.
ShowFlags	Retorna y establece la bandera de visibilidad de los elementos del gráfico.
IsShowLegend	Retorna la bandera de visibilidad de la leyenda en el gráfico.
IsShowScaleLeft	Retorna la bandera de visibilidad de la escala de valores de la izquierda.

© 2000-2025, MetaQuotes Ltd.

Método	Acción
IsShowScaleRight	Retorna la bandera de visibilidad de la escala de valores de la derecha.
IsShowScaleTop	Retorna la bandera de visibilidad de la escala de valores de arriba.
IsShowScaleBottom	Retorna la bandera de visibilidad de la escala de valores de abajo.
IsShowGrid	Retorna la bandera de visibilidad de la cuadrícula en el gráfico.
IsShowDescriptors	Retorna la bandera de visibilidad de los descriptores en el gráfico.
IsShowPercent	Retorna la bandera de visibilidad del tanto por ciento en el gráfico.
VScaleMin	Retorna y establece el mínimo en la escala vertical de valores.
VScaleMax	Retorna y establece el máximo en la escala vertical de valores.
NumGrid	Retorna y establece el número de secciones verticales de la escala al construir la cuadrícula del gráfico.
DataOffset	Retorna y establece el valor del desplazamiento de los datos.
DataTotal	Retorna el número total de series de datos en el gráfico.
DrawDescriptors	Método virtual para dibujar los descriptores.
DrawData	Método virtual para dibujar una serie de datos según el índice especificado.
Create	Método virtual que crea un recurso gráfico.
AllowedShowFlags	Establece el conjunto de banderas de visibilidad permitidas para los elementos del gráfico.
ShowLegend	Establece la bandera de visibilidad de la leyenda.
ShowScaleLeft	Establece la bandera de visibilidad para la escala izquierda.
ShowScaleRight	Establece la bandera de visibilidad para la escala derecha.

© 2000-2025, MetaQuotes Ltd.

Método	Acción
ShowScaleTop	Establece la bandera de visibilidad para la escala superior.
ShowScaleBottom	Establece la bandera de visibilidad para la escala inferior.
ShowGrid	Establece la bandera de visibilidad para la cuadrícula.
ShowDescriptors	Establece la bandera de visibilidad para los descriptores.
ShowValue	Establece la bandera de visibilidad para los valores.
ShowPercent	Establece la bandera de visibilidad para los tantos por ciento.
LegendAlignment	Establece la alineación del texto para la leyenda.
Accumulative	Establece la bandera de acumulación de valores para la serie.
VScaleParams	Establece los parámetros para la escala vertical de valores.
DescriptorUpdate	Actualiza el valor del descriptor de la serie (según la posición indicada).
ColorUpdate	Actualiza los colores de la serie (según la posición indicada).
ValuesCheck	Realiza el cálculo interno para la construcción del gráfico.
Redraw	Redibuja el gráfico.
DrawBackground	Dibuja el fondo.
DrawLegend	Redibuja la leyenda.
DrawLegendVertical	Dibuja la leyenda vertical.
DrawLegendHorizontal	Dibuja la leyenda horizontal.
CalcScales	Calcula las coordenadas de la escala.
DrawScales	Redibuja todas las escalas de valores.
DrawScaleLeft	Redibuja la escala de valores de la izquierda.
DrawScaleRight	Redibuja la escala de valores de la derecha.
DrawScaleTop	Redibuja la escala de valores superior.

© 2000-2025, MetaQuotes Ltd.

Método	Acción
DrawScaleBottom	Redibuja la escala de valores inferior.
DrawGrid	Redibuja la cuadrícula.
DrawChart	Redibuja los datos.

Métodos heredados de la clase CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

© 2000-2025, MetaQuotes Ltd.

ColorBackground (método Get)

Devuelve el color del fondo.

```
uint  ColorBackground()
```

Valor devuelto

Color del fondo.

ColorBackground (método Set)

Establece el color del fondo trasero.

```
void  ColorBackground(
  const uint  value,  // color del fondo
  )
```

Parámetros

value

[in] Color del fondo.

© 2000-2025, MetaQuotes Ltd.

ColorBorder (método Get)

Retorna el color de los límites.

```
uint  ColorBorder()
```

Valor devuelto

Color de los límites.

ColorBorder (método Set)

Establece el color de los límites

```
void  ColorBorder(
  const uint  value,  // color de los límites
  )
```

Parámetros

value

[in] Color de los límites.

© 2000-2025, MetaQuotes Ltd.

ColorText (método Get)

Retorna el color del texto.

```
uint  ColorText()
```

Valor devuelto

Color del texto.

ColorText (método Set)

Establece el color del texto.

```
void  ColorText(
  const uint  value,  // color del texto
  )
```

Parámetros

value

[in] Color del texto.

© 2000-2025, MetaQuotes Ltd.

ColorGrid (método Get)

Retorna el color de la cuadrícula.

```
uint  ColorGrid()
```

Valor devuelto

Color de la cuadrícula.

ColorGrid (método Set)

Establece el color de la cuadrícula.

```
void  ColorGrid(
  const uint  value,  // color de la cuadrícula
  )
```

Parámetros

value

[in] Color de la cuadrícula.

© 2000-2025, MetaQuotes Ltd.

MaxData (método Get)

Retorna el número máximo de datos permitido (series).

```
uint  MaxData()
```

Valor devuelto

Número máximo de datos (series).

MaxData (método Set)

Establece el número máximo de datos permitido (series).

```
void  MaxData(
  const uint  value,  // Número de datos
  )
```

Parámetros

value

[in] Número máximo de datos (series).

© 2000-2025, MetaQuotes Ltd.

MaxDescrLen (método Get)

Retorna la longitud máxima de los descriptores.

```
uint  MaxDescrLen()
```

Valor devuelto

Valor de la longitud máxima de los descriptores.

MaxDescrLen (método Set)

Establece la longitud máxima de los descriptores.

```
void  MaxDescrLen(
  const uint  value,  // longitud máxima
  )
```

Parámetros

value

[in] Valor de la longitud máxima de los descriptores.

© 2000-2025, MetaQuotes Ltd.

ShowFlags (método Get)

Retorna la bandera de visibilidad de los elementos del gráfico.

```
bool  ShowFlags()
```

Valor devuelto

Valor de la bandera de visibilidad de los elementos del gráfico.

ShowFlags (método Set)

Establece la bandera de visibilidad de los elementos del gráfico.

```
void  ShowFlags(
  const uint  flags,  // bandera
  )
```

Parameters

flags

[in] Valor de la bandera de visibilidad de los elementos del gráfico.

© 2000-2025, MetaQuotes Ltd.

IsShowLegend

Retorna la bandera de visibilidad de la leyenda en el gráfico.

```
bool  IsShowLegend()
```

Valor devuelto

true si la leyenda es visible, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleLeft

Retorna la bandera de visibilidad de la escala de valores de la izquierda.

```
bool  IsShowScaleLeft()
```

Valor devuelto

true si la escala de valores es visible, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleRight

Retorna la bandera de visibilidad de la escala de valores de la derecha.

```
bool  IsShowScaleRight()
```

Valor devuelto

true si la escala de valores es visible, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleTop

Retorna la bandera de visibilidad de la escala de valores de arriba.

```
bool  IsShowScaleTop()
```

Valor devuelto

true si la escala de valores es visible, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleBottom

Retorna la bandera de visibilidad de la escala de valores de abajo.

```
bool  IsShowScaleBottom()
```

Valor devuelto

true si la escala de valores es visible, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsShowGrid

Retorna la bandera de visibilidad de la cuadrícula en el gráfico.

```
bool  IsShowGrid()
```

Valor devuelto

true si la cruceta es visible, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsShowDescriptors

Retorna la bandera de visibilidad de los descriptores en el gráfico.

```
bool  IsShowDescriptors()
```

Valor devuelto

true si los descriptores son visibles, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

IsShowPercent

Retorna la bandera de visibilidad del tanto por ciento en el gráfico.

```
bool  IsShowPercent()
```

Valor devuelto

true si los tantos por ciento son visibles, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

VScaleMin (Valor Get)

Retorna el mínimo en la escala vertical de valores.

```
double  VScaleMin()
```

Valor devuelto

Valor mínimo en la escala vertical.

VScaleMin (Método Set)

Establece el mínimo en la escala vertical de valores.

```
void  VScaleMin(
  const double  value,    // valor en la escala vertical
  )
```

Parámetros

value

[in] Valor mínimo.

© 2000-2025, MetaQuotes Ltd.

VScaleMax

Retorna el máximo en la escala vertical de valores.

```
double  VScaleMax()
```

Valor devuelto

Valor máximo en la escala vertical.

VScaleMax

Establece el máximo en la escala vertical de valores.

```
void  VScaleMax(
  const double  value,  //   valor en la vertical
  )
```

Parámetros

value

[in] Valor mínimo.

© 2000-2025, MetaQuotes Ltd.

NumGrid

Retorna el número de secciones verticales de la escala al construir la cuadrícula del gráfico.

```
uint  NumGrid()
```

Valor devuelto

Número de secciones.

NumGrid

Establece el número de secciones verticales de la escala al construir la cuadrícula del gráfico.

```
void  NumGrid(
  const uint  value,  // número secciones
  )
```

Parámetros

value

[in] Número de secciones.

© 2000-2025, MetaQuotes Ltd.

DataOffset

Retorna el valor de desplazamiento de los datos.

```
int  DataOffset()
```

Valor devuelto

Desplazamiento de los datos.

DataOffset

Establece el valor del desplazamiento de los datos.

```
void  DataOffset(
  const int  value,  // valor
  )
```

Parámetros

value

[in] Desplazamiento de los datos.

© 2000-2025, MetaQuotes Ltd.

DataTotal

Retorna el número total de series de datos en el gráfico.

```
uint  DataTotal()
```

Valor devuelto

Número de series.

© 2000-2025, MetaQuotes Ltd.

DrawDescriptors

Método virtual para dibujar los descriptores.

```
virtual void  DrawDescriptors()
```

© 2000-2025, MetaQuotes Ltd.

DrawData

Método virtual para dibujar una serie de datos según el índice especificado.

```
virtual void  DrawData(
   const uint  idx=0,  // índice
   )
```

Parámetros

idx=0

[in] Índice de la serie.

© 2000-2025, MetaQuotes Ltd.

Create

Método virtual que crea un recurso gráfico.

```
virtual bool  Create(
  const string       name,      // nombre del recurso
  const int          width,     // anchura
  const int          height,    // altura
  ENUM_COLOR_FORMAT  clrfmt,    // formato
  )
```

Parámetros

name

[in] Base para el nombre del recurso gráfico. El nombre del recurso se forma añadiendo una línea pseudoaleatoria.

width

[in] Anchura (tamaño en el eje X) en píxeles.

height

[in] Altura (tamaño en el eje Y) en píxeles.

clrfmt

[in] Método de procesamiento del color. Podrá leer información más detallada sobre los métodos de procesamiento del color en la descripción de la función ResourceCreate().

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

AllowedShowFlags

Establece el conjunto de banderas de visibilidad permitidas para los elementos del gráfico.

```
void  AllowedShowFlags(
  const uint  flags,  // banderas
  )
```

Parámetros

flags

[in] Banderas permitidas.

© 2000-2025, MetaQuotes Ltd.

ShowLegend

Establece el valor de la bandera de visibilidad para la leyenda (FLAG_SHOW_LEGEND).

```
void  ShowLegend(
  const bool  flag,  // valor de la bandera
  )
```

Parámetros

flag

[in] Valor de la bandera:

- true — la leyenda se hace visible.
- false — la leyenda se hace invisible.

© 2000-2025, MetaQuotes Ltd.

ShowScaleLeft

Establece el valor de la bandera de visibilidad para la escala izquierda (FLAG_SHOW_SCALE_LEFT).

```
void  ShowScaleLeft(
  const bool  flag,  // valor de la bandera
  )
```

Parámetros

flag

[in] Valor de la bandera:

- true — la escala izquierda se hace visible.
- false — la escala izquierda se hace invisible.

© 2000-2025, MetaQuotes Ltd.

ShowScaleRight

Establece el valor de la bandera de visibilidad para la escala derecha (FLAG_SHOW_SCALE_RIGHT).

```
void  ShowScaleRight(
  const bool  flag,  // valor de la bandera
  )
```

Parámetros

flag

[in] Valor de la bandera:

- true — la escala derecha se hace visible.
- false — la escala derecha se hace invisible.

© 2000-2025, MetaQuotes Ltd.

ShowScaleTop

Establece el valor de la bandera de visibilidad para la escala superior (FLAG_SHOW_SCALE_TOP).

```
void  ShowScaleTop(
  const bool  flag,  // valor de la bandera
  )
```

Parámetros

flag

[in] Valor de la bandera:

- true — la escala superior se hace visible.
- false — la escala superior se hace invisible.

© 2000-2025, MetaQuotes Ltd.

ShowScaleBottom

Establece el valor de la bandera de visibilidad para la escala inferior (FLAG_SHOW_SCALE_BOTTOM).

```
void  ShowScaleBottom(
  const bool  flag,     // valor de la bandera
  )
```

Parámetros

flag

[in] Valor de la bandera:

- true — la escala inferior se hace visible.
- false — la escala inferior se hace invisible.

© 2000-2025, MetaQuotes Ltd.

ShowGrid

Establece el valor de la bandera de visibilidad para la cuadrícula (FLAG_SHOW_GRID).

```
void  ShowGrid(
  const bool  flag,  // valor de la bandera
  )
```

Parámetros

flag

[in] Valor de la bandera:

- true — la cuadrícula se hace visible.
- false — la cuadrícula se hace invisible.

© 2000-2025, MetaQuotes Ltd.

ShowDescriptors

Establece el valor de la bandera de visibilidad para los descriptores (FLAG_SHOW_DESCRIPTORS).

```
void  ShowDescriptors(
  const bool  flag,    // valor de la bandera
  )
```

Parámetros

flag

[in] Valor de la bandera:

- true — el descriptor se hace visible.
- false — el descriptor se hace invisible.

© 2000-2025, MetaQuotes Ltd.

ShowValue

Establece la bandera de visibilidad para los valores (FLAG_SHOW_VALUE).

```
void  ShowValue(
  const bool  flag,  // valor de la bandera
  )
```

Parámetros

flag

[in] Valor de la bandera:

- true — los valores se hacen visibles.
- false — los valores se hacen invisibles.

© 2000-2025, MetaQuotes Ltd.

ShowPercent

Establece el valor de la bandera de visibilidad para los tantos por ciento (FLAG_SHOW_PERCENT).

```
void  ShowPercent(
  const bool  flag,  // valor de la bandera
  )
```

Parámetros

flag

[in] Valor de la bandera:

- true — el tanto por ciento se hace visible.
- false — el tanto por ciento se hace invisible.

© 2000-2025, MetaQuotes Ltd.

LegendAlignment

Establece la alineación del texto para la leyenda.

```
void  LegendAlignment(
  const ENUM_ALIGNMENT  value,  // bandera
  )
```

Parámetros

value

[in] Adopta uno de los valores de la enumeración ENUM_ALIGNMENT:

- ALIGNMENT_LEFT — alinea por el límite izquierdo.
- ALIGNMENT_TOP — alinea por el límite superior.
- ALIGNMENT_RIGHT — alinea por el límite derecho.
- ALIGNMENT_BOTTOM — alinea por el límite inferior.

© 2000-2025, MetaQuotes Ltd.

Accumulative

Establece la bandera de acumulación de valores para la serie.

```
void  Accumulative(
  const bool  flag=true,  // valor de la bandera
  )
```

Parámetros

flag=true

[in] Valor de la bandera:

- true — el valor actual de la serie se sustituye con la suma de todos los anteriores.
- false — mod estándar de dibujado de la serie.

© 2000-2025, MetaQuotes Ltd.

VScaleParams

Establece los parámetros para la escala vertical de valores.

```
void  VScaleParams(
  const double  max,   // máximo
  const double  min,   // mínimo
  const uint    grid,  // número de secciones
  )
```

Parámetros

max

[in] Valor mínimo.

min

[in] Valor máximo.

grid

[in] Número de secciones en la escala.

© 2000-2025, MetaQuotes Ltd.

DescriptorUpdate

Actualiza el valor del descriptor de la serie (según la posición indicada).

```
bool  DescriptorUpdate(
  const uint    pos,    // índice
  const string  descr,  // valor
  )
```

Parámetros

pos

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir del 0.

descr

[in] Valor del descriptor.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ColorUpdate

Actualiza los colores de la serie (según la posición indicada).

```
bool  ColorUpdate(
  const uint  pos,  // índice
  const uint  clr,  // color
  )
```

Parámetros

pos

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir del 0.

clr

[in] Valor del color.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ValuesCheck

Método virtual auxiliar que realiza el cálculo interno para la construcción del gráfico.

```
virtual void  ValuesCheck()
```

© 2000-2025, MetaQuotes Ltd.

Redraw

Método virtual para el redibujado del gráfico.

```
virtual void  Redraw()
```

© 2000-2025, MetaQuotes Ltd.

DrawBackground

Método virtual para el redibujado del fondo.

```
virtual void  DrawBackground()
```

© 2000-2025, MetaQuotes Ltd.

DrawLegend

Método virtual para el redibujado de la leyenda.

```
virtual void  DrawLegend()
```

© 2000-2025, MetaQuotes Ltd.

DrawLegendVertical

Dibuja la leyenda vertical.

```
int  DrawLegendVertical(
  const int  w,  // anchura
  const int  h,  // altura
  )
```

Parámetros

w

[in] Anchura máxima del texto en la leyenda.

h

[in] Altura máxima del texto en la leyenda.

Valor devuelto

Anchura de la leyenda en píxeles.

© 2000-2025, MetaQuotes Ltd.

DrawLegendHorizontal

Dibuja la leyenda horizontal.

```
int  DrawLegendHorizontal(
  const int  w,  //
  const int  h,  //
  )
```

Parámetros

w

[in] Anchura máxima del texto en la leyenda.

h

[in] Altura máxima del texto en la leyenda.

Valor devuelto

Altura de la leyenda en píxeles.

© 2000-2025, MetaQuotes Ltd.

CalcScales

Método virtual para calcular las coordenadas de los rótulos para la escala de valores.

```
virtual void  CalcScales()
```

© 2000-2025, MetaQuotes Ltd.

DrawScales

Método virtual para el redibujado de todas las escalas de valores.

```
virtual void  DrawScales()
```

© 2000-2025, MetaQuotes Ltd.

DrawScaleLeft

Método virtual para el redibujado de la escala izquierda de valores.

```
virtual int  DrawScaleLeft(
  const bool  draw,  // bandera
  )
```

Parámetros

draw

[in] bandera que indica si hay que redibujar la escala.

Valor devuelto

Anchura de la escala de valores.

© 2000-2025, MetaQuotes Ltd.

DrawScaleRight

Método virtual para el redibujado de la escala derecha de valores.

```
virtual int  DrawScaleRight(
  const bool  draw,  // bandera
  )
```

Parámetros

draw

[in] bandera que indica si hay que redibujar la escala.

Valor devuelto

Anchura de la escala de valores.

© 2000-2025, MetaQuotes Ltd.

DrawScaleTop

Método virtual para el redibujado de la escala superior de valores.

```
virtual int  DrawScaleTop(
  const bool  draw,  // bandera
  )
```

Parámetros

draw

[in] bandera que indica si hay que redibujar la escala.

Valor devuelto

Altura de la escala de valores.

© 2000-2025, MetaQuotes Ltd.

DrawScaleBottom

Método virtual para el redibujado de la escala inferior de valores.

```
virtual int  DrawScaleBottom(
  const bool  draw,  // bandera
  )
```

Parámetros

draw

[in] bandera que indica si hay que redibujar la escala.

Valor devuelto

Altura de la escala de valores.

© 2000-2025, MetaQuotes Ltd.

DrawGrid

Método virtual para el redibujado de la cuadrícula.

```
virtual void  DrawGrid()
```

© 2000-2025, MetaQuotes Ltd.

DrawChart

Método virtual para el redibujado del gráfico.

```
virtual void  DrawChart()
```

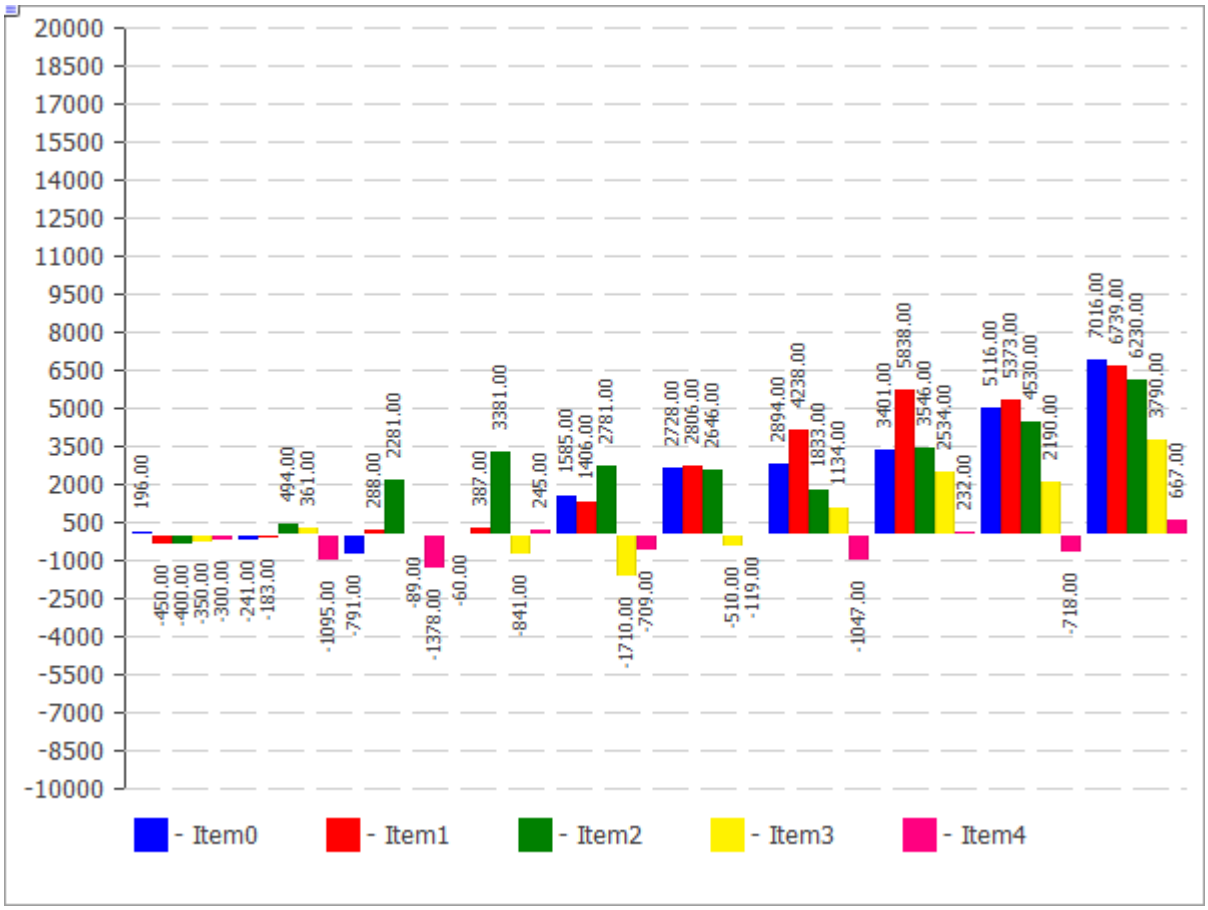
© 2000-2025, MetaQuotes Ltd.

CHistogramChart

Clase para construir histogramas.

Descripción

En esta clase se han implementado todos los métodos de trabajo con construcción de histogramas de columnas. Con su ayuda se establece la anchura de las columnas, se configura el trabajo con las series de datos. Se han incluido los métodos de trabajo con coloreado en gradiente de las columnas del histograma, lo que permite representar la imagen de los datos de forma más visual.



El código de la figura mostrada más arriba se muestra abajo.

Declaración

```
class CHistogramChart : public CChartCanvas
```

Encabezamiento

```
#include <Canvas\Charts\HistogramChart.mqh>
```

Jerarquía de herencia

CCanvas
  CChartCanvas
    CHistogramChart

© 2000-2025, MetaQuotes Ltd.

Métodos de clase

Método	Acción
Gradient	Establece la bandera que indica si se va a aplicar el coloreado en gradiente de las columnas del histograma.
BarGap	Establece el valor del margen del histograma partiendo del comienzo de las coordenadas.
BarMinSize	Establece la anchura mínima de las columnas del histograma.
BarBorder	Establece la bandera que indica la necesidad de dibujar el límite para cada columna.
Create	Método virtual que crea un recurso gráfico.
SeriesAdd	Añade una nueva serie de datos.
SeriesInsert	Inserta una serie de datos en el gráfico.
SeriesUpdate	Actualiza una serie de datos en el gráfico.
SeriesDelete	Elimina una serie de datos en el gráfico.
ValueUpdate	Actualiza el valor de un elemento en la serie indicada.
DrawData	Método virtual que dibuja un histograma para la serie indicada.
DrawBar	Dibuja una columna del histograma en forma de rectángulo coloreado.
GradientBrush	Crea un pincel para el coloreado en gradiente.

Métodos heredados de la clase CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Métodos heredados de la clase CChartCanvas

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

Ejemplo

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          HistogramChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Example of using histogram"
//---
#include <Canvas\Charts\HistogramChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=true;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- create chart
   CHistogramChart chart;
   if(!chart.CreateBitmapLabel("SampleHistogramChart",10,10,600,450))
     {
      Print("Error creating histogram chart: ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,20);
   chart.ShowValue(true);
   chart.ShowScaleTop(false);
   chart.ShowScaleBottom(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- play with values
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- finish
   chart.Destroy();
   return(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

Gradient

Establece la bandera que indica si se va a aplicar el coloreado en gradiente de las columnas del histograma.

```
void  Gradient(
  const bool  flag=true,  // valor de la bandera
  )
```

Parámetros

flag=true

Valor de la bandera: true, si está activado el coloreado en gradiente, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

BarGap

Establece el valor del margen del histograma partiendo del comienzo de las coordenadas.

```
void  BarGap(
  const uint  value,  // margen
  )
```

Parámetros

value

[in] Valor del margen del histograma.

© 2000-2025, MetaQuotes Ltd.

BarMinSize

Establece la anchura mínima de las columnas del histograma.

```
void  BarMinSize(
  const uint  value,  // anchura mínima
  )
```

Parámetros

value

[in] Anchura mínima.

© 2000-2025, MetaQuotes Ltd.

BarBorder

Establece la bandera que indica la necesidad de dibujar el límite para cada columna.

```
void  BarBorder(
  const uint  value,  // bandera
  )
```

Parámetros

value

[in] Valor de la bandera:

- true — los límites se dibujarán
- false — los límites no se dibujarán

© 2000-2025, MetaQuotes Ltd.

Create

Método virtual que crea un recurso gráfico.

```
virtual bool  Create(
  const string       name,    // nombre
  const int          width,   // anchura
  const int          height,  // altura
  ENUM_COLOR_FORMAT  clrfmt,  // formato
  )
```

Parámetros

name

[in] Base para el nombre del recurso gráfico. El nombre del recurso se forma añadiendo una línea pseudoaleatoria.

width

[in] Anchura (tamaño en el eje X) en píxeles.

height

[in] Altura (tamaño en el eje Y) en píxeles.

clrfmt

[in] Método de procesamiento del color. Podrá leer información más detallada sobre los métodos de procesamiento del color en la descripción de la función ResourceCreate().

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SeriesAdd

Añade una nueva serie de datos.

```
bool  SeriesAdd(
  const double&  value[],  // valores
  const string   descr,    // rótulo
  const uint     clr,      // color
  )
```

Parámetros

value[]

[in] Serie de datos.

descr

[in] Rótulo de la serie.

clr

[in] Color de representación de la serie.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SeriesInsert

Inserta una serie de datos en el gráfico.

```
bool  SeriesInsert(
  const uint    pos,        // índice
  const double&  value[],   // valores
  const string  descr,      // rótulo
  const uint    clr,        // color
  )
```

Parámetros

pos

[in] Índice para la inserción.

value[]

[in] Serie de datos.

descr

[in] Rótulo de la serie.

clr

[in] Color de representación de la serie.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SeriesUpdate

Actualiza una serie de datos en el gráfico.

```
bool  SeriesUpdate(
  const uint    pos,        // índice
  const double  &value[],   // valores
  const string  descr,      // rótulo
  const uint    clr,        // color
  )
```

Parámetros

pos

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir de 0.

&value[]

[in] Nuevos valores para la serie de datos.

descr

[in] Rótulo de la serie.

clr

[in] Color de representación de la serie.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SeriesDelete

Elimina una serie de datos en el gráfico.

```
bool  SeriesDelete(
  const uint  pos,  // índice de la serie
  )
```

Parámetros

pos

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir del 0.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ValueUpdate

Actualiza el valor indicado en la serie indicada.

```
bool  ValueUpdate(
  const uint  series,  // índice de la serie
  const uint  pos,     // índice del elemento
  double      value,   // valor
  )
```

Parámetros

series

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir del 0.

pos

[in] Índice del elemento en la serie.

value

[in] Nuevo valor.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

DrawData

Método virtual que dibuja un histograma para la serie indicada.

```
virtual void  DrawData(
  const uint  index,  // índice
  )
```

Parámetros

index

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir de 0.

© 2000-2025, MetaQuotes Ltd.

DrawBar

Dibuja una columna del histograma en forma de rectángulo coloreado.

```
void  DrawBar(
  const int   x,    // coordenada x
  const int   y,    // coordenada y
  const int   w,    // anchura
  const int   h,    // altura
  const uint  clr,  // color
  )
```

Parámetros

x

[in] Coordenada X del punto superior izquierdo del rectángulo.

y

[in] Coordenada Y del punto superior izquierdo del rectángulo.

w

[in] Anchura del rectángulo.

h

[in] Altura del rectángulo.

clr

[in] Color del rectángulo.

© 2000-2025, MetaQuotes Ltd.

GradientBrush

Crea un pincel para el coloreado en gradiente.

```
void  GradientBrush(
  const int   size,      // tamaño
  const uint  fill_clr,  // color del coloreado
  )
```

Parámetros

size

[in] Grosor del pincel

fill_clr

[in] Color del coloreado

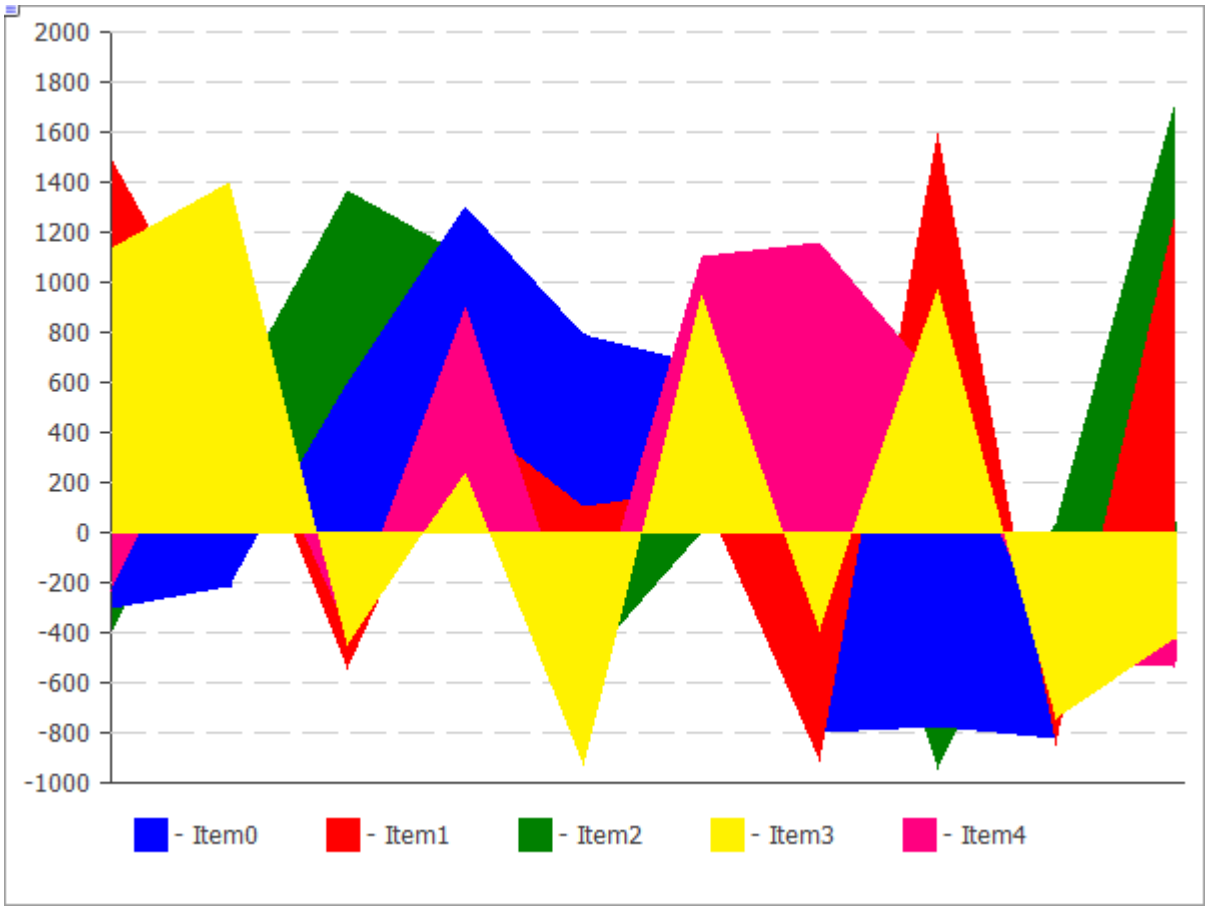
© 2000-2025, MetaQuotes Ltd.

CLineChart

Clase para la construcción de curvas.

Descripción

Los métodos incluidos en la clase están diseñados para trabajar con las curvas en el gráfico. Se ha incluido la posibilidad de colorear la zona limitada con la curva construida.



El código de la figura mostrada más arriba se muestra abajo.

Declaración

```
class CLineChart : public CChartCanvas
```

Encabezamiento

```
#include <Canvas\Charts\LineChart.mqh>
```

Jerarquía de herencia

CCanvas
  CChartCanvas
    CLineChart

Métodos de clase

© 2000-2025, MetaQuotes Ltd.

Method	Action
Filled	Establece la bandera de coloreado de la zona debajo de la curva definida por la serie de datos.
Create	Crea un recurso gráfico.
SeriesAdd	Añade una nueva serie de datos.
SeriesInsert	Inserta una serie de datos en el gráfico.
SeriesUpdate	Actualiza una serie de datos en el gráfico.
SeriesDelete	Elimina una serie de datos del gráfico.
ValueUpdate	Actualiza el valor indicado en la serie indicada.
DrawChart	Método virtual que ejecuta el dibujado de la curva y de todos sus elementos.
DrawData	Método virtual que dibuja la curva para la serie definida.
CalcArea	Calcula la superficie debajo de la curva definida por la serie de datos.

Métodos heredados de la clase CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Métodos heredados de la clase CChartCanvas

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

Ejemplo

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                LineChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Example of using line chart"
//---
#include <Canvas\Charts\LineChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=false;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- create chart
   CLineChart chart;
//--- create chart
   if(!chart.CreateBitmapLabel("SampleHistogrammChart",10,10,600,450))
     {
      Print("Error creating line chart: ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,15);
   chart.ShowScaleTop(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   chart.Filled();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- play with values
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- finish
   chart.Destroy();
   return(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

Filled

Establece la bandera que indica si hay que colorear la zona debajo de la curva definida con la serie de datos.

```
void  Filled(
  const bool  flag=true,  // bandera
  )
```

Parámetros

flag=true

[in] Valor de la bandera:

- true — colorear la zona bajo la curva
- false — no colorear la zona bajo la curva

© 2000-2025, MetaQuotes Ltd.

Create

Método virtual que crea un recurso gráfico.

```
virtual bool  Create(
  const string       name,    // nombre
  const int          width,   // anchura
  const int          height,  // altura
  ENUM_COLOR_FORMAT  clrfmt,  // formato
  )
```

Parámetros

name

[in] Base para el nombre del recurso gráfico. El nombre del recurso se forma añadiendo una línea pseudoaleatoria.

width

[in] Anchura (tamaño en el eje X) en píxeles.

height

[in] Altura (tamaño en el eje Y) en píxeles.

clrfmt

[in] Método de procesamiento del color. Podrá leer información más detallada sobre los métodos de procesamiento del color en la descripción de la función ResourceCreate().

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SeriesAdd

Añade una nueva serie de datos.

```
bool  SeriesAdd(
  const double&  value[],  // valores
  const string   descr,    // rótulo
  const uint     clr,      // color
  )
```

Parámetros

value[]

[in] Serie de datos.

descr

[in] Rótulo de la serie.

clr

[in] Color de representación de la serie.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SeriesInsert

Inserta una serie de datos en el gráfico.

```
bool  SeriesInsert(
  const uint     pos,        // índice
  const double&  value[],    // valores
  const string   descr,      // rótulo
  const uint     clr,        // color
  )
```

Parámetros

pos

[in] Índice para la inserción.

value[]

[in] Serie de datos.

descr

[in] Rótulo de la serie.

clr

[in] Color de representación de la serie.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SeriesUpdate

Actualiza una serie de datos en el gráfico.

```
bool  SeriesUpdate(
  const uint     pos,        // índice
  const double&  value[],    // valores
  const string   descr,      // rótulo
  const uint     clr,        // color
  )
```

Parámetros

pos

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir de 0.

value[]

[in] Nuevos valores para la serie de datos.

descr

[in] Rótulo de la serie.

clr

[in] Color de representación de la serie.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SeriesDelete

Elimina una serie de datos del gráfico.

```
bool  SeriesDelete(
  const uint  pos,  // índice de la serie
  )
```

Parámetros

pos

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir de 0.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ValueUpdate

Actualiza el valor indicado en la serie indicada.

```
bool  ValueUpdate(
  const uint  series,  // índice de la serie
  const uint  pos,     // índice del elemento
  double      value,   // valor
  )
```

Parámetros

series

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir del 0.

pos

[in] Índice del elemento en la serie.

value

[in] Nuevo valor.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

DrawChart

Método virtual que ejecuta el dibujado de la curva y de todos sus elementos.

```
virtual void  DrawChart()
```

© 2000-2025, MetaQuotes Ltd.

DrawData

Método virtual que dibuja la curva para la serie definida.

```
virtual void  DrawData(
  const uint  index,  // índice
  )
```

Parameters

index

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir de 0.

© 2000-2025, MetaQuotes Ltd.

CalcArea

Calcula la superficie debajo de la curva definida por la serie de datos.

```
double  CalcArea(
  const uint  index,  // índice
  )
```

Parámetros

index

[in] El índice de la serie supone el número ordinal de su adición, comenzando a partir del 0.

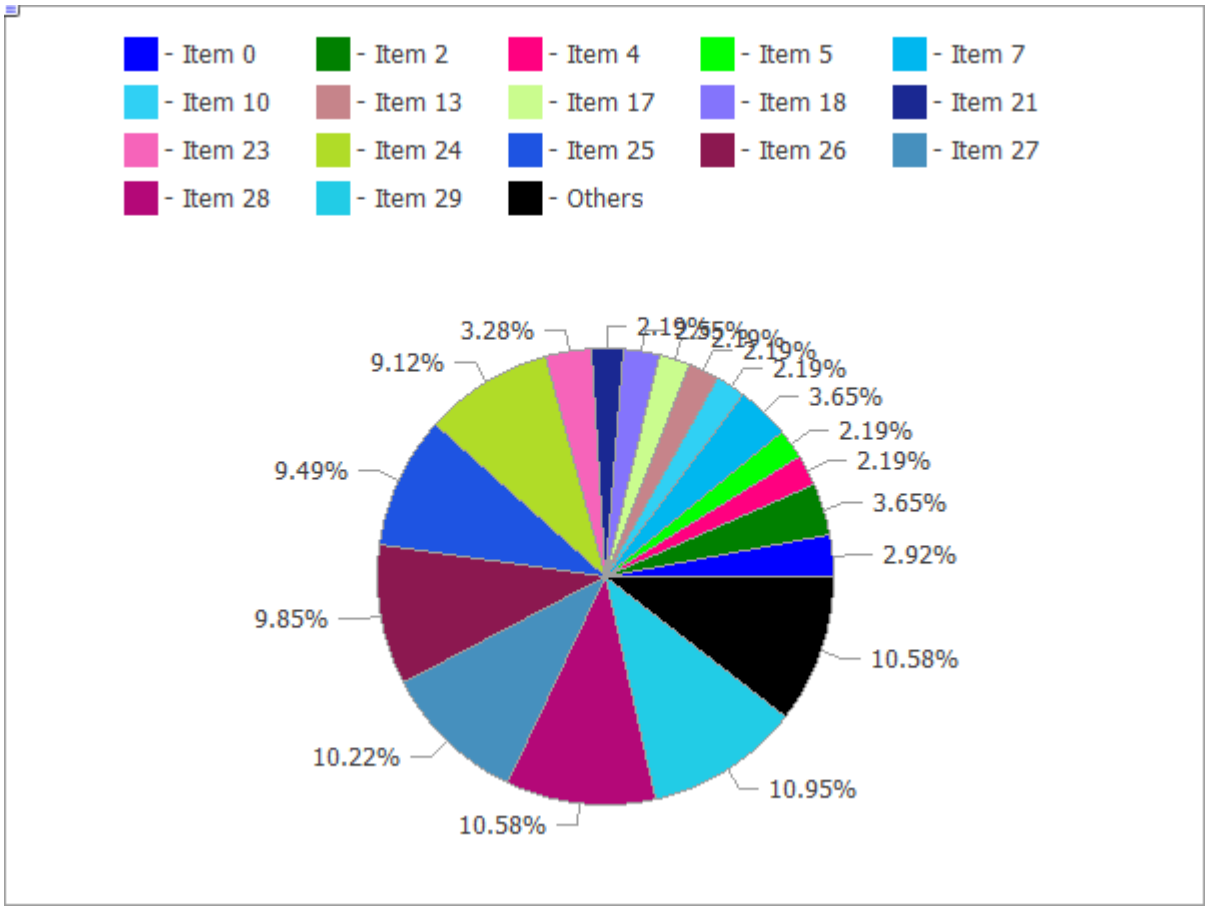
Valor devuelto

Superficie de la figura limitada por la curva definida con la serie de datos.

© 2000-2025, MetaQuotes Ltd.

CPieChart

Clase para construir diagramas circulares.



El código de la figura mostrada más arriba se muestra abajo.

Descripción

Los métodos incluidos en esta clase han sido diseñados para lograr un trabajo completo con los diagramas circulares, empezando por la creación de un recurso gráfico y finalizando con la presentación de los rótulos de los segmentos.

Declaración

```
class CPieChart : public CChartCanvas
```

Encabezamiento

```
#include <Canvas\Charts\PieChart.mqh>
```

Jerarquía de herencia

CCanvas

CChartCanvas

CPieChart

© 2000-2025, MetaQuotes Ltd.

Métodos de clase

Método	Acción
Create	Método virtual que crea un recurso gráfico.
SeriesSet	Establece la serie de valores que se mostrará en el diagrama.
ValueAdd	Añade un nuevo valor al diagrama (al final).
ValueInsert	Inserta un nuevo valor en el diagrama (según la posición indicada).
ValueUpdate	Actualiza los valores en el diagrama (según la posición indicada).
ValueDelete	Elimina un valor del diagrama (según la posición indicada).
DrawChart	Método virtual que ejecuta el dibujado del diagrama y de todos sus elementos.
DrawPie	Dibuja un segmento del diagrama que se corresponde con un valor determinado.
LabelMake	Genera el rótulo de un segmento basándose en sus valores y su rótulo inicial.

Métodos heredados de la clase CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Métodos heredados de la clase CChartCanvas

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

© 2000-2025, MetaQuotes Ltd.

Ejemplo

```
//+------------------------------------------------------------------+
//|                                                PieChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Example of using pie chart"
//---
#include <Canvas\Charts\PieChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input int      Width=600;
input int      Height=450;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
//--- check
   if(Width<=0 || Height<=0)
     {
      Print("Too simple.");
      return(-1);
     }
//--- create chart
   CPieChart pie_chart;
   if(!pie_chart.CreateBitmapLabel("PieChart",10,10,Width,Height))
     {
      Print("Error creating pie chart: ",GetLastError());
      return(-1);
     }
   pie_chart.ShowPercent();
//--- draw
   for(uint i=0;i<30;i++)
     {
      pie_chart.ValueAdd(100*(i+1),"Item "+IntegerToString(i));
      Sleep(10);
     }
   Sleep(2000);
//--- disable legend
   pie_chart.LegendAlignment(ALIGNMENT_LEFT);
   Sleep(2000);
//--- disable legend
   pie_chart.LegendAlignment(ALIGNMENT_RIGHT);
   Sleep(2000);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- disable legend
   pie_chart.LegendAlignment(ALIGNMENT_TOP);
   Sleep(2000);
//--- disable legend
   pie_chart.ShowLegend(false);
   Sleep(2000);
//--- disable percentage
   pie_chart.ShowPercent(false);
   Sleep(2000);
//--- disable descriptors
   pie_chart.ShowDescriptors(false);
   Sleep(2000);
//--- enable all
   pie_chart.ShowLegend();
   pie_chart.ShowValue();
   pie_chart.ShowDescriptors();
   Sleep(2000);
//--- or like this
   pie_chart.ShowFlags(FLAG_SHOW_LEGEND|FLAG_SHOW_DESCRIPTORS|FLAG_SHOW_PERCENT);
   uint total=pie_chart.DataTotal();
//--- play with values
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ValueUpdate(i,100*(rand()%10+1));
      Sleep(1000);
     }
//--- play with colors
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ColorUpdate(i%total,RandomRGB());
      Sleep(1000);
     }
//--- rotate
   while(!IsStopped())
     {
      pie_chart.DataOffset(pie_chart.DataOffset()+1);
      Sleep(200);
     }
//--- finish
   pie_chart.Destroy();
   return(0);
  }
//+------------------------------------------------------------------+
//| Random RGB color                                                 |
//+------------------------------------------------------------------+
uint RandomRGB(void)
  {
   return(XRGB(rand()%255,rand()%255,rand()%255));
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Método virtual que crea un recurso gráfico.

```
virtual bool  Create(
  const string       name,    // nombre
  const int          width,   // anchura
  const int          height,  // altura
  ENUM_COLOR_FORMAT  clrfmt,  // formato
  )
```

Parámetros

name

[in] Base para el nombre del recurso gráfico. El nombre del recurso se forma añadiendo una línea pseudoaleatoria.

width

[in] Anchura (tamaño en el eje X) en píxeles.

height

[in] Altura (tamaño en el eje Y) en píxeles.

clrfmt

[in] Método de procesamiento del color. Podrá leer información más detallada sobre los métodos de procesamiento del color en la descripción de la función ResourceCreate().

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

SeriesSet

Establece la serie de valores que se mostrará en el diagrama.

```
bool  SeriesSet(
  const double&  value[],  // valores
  const string&  text[],   // rótulos
  const uint&    clr[],    // colores
  )
```

Parámetros

value[]

[in] Matriz de valores.

text[]

[in] Matriz de los rótulos de los valores.

clr[]

[in] Matriz de los colores de los valores.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ValueAdd

Añade un nuevo valor al diagrama (al final).

```
bool  ValueAdd(
  const double  value,  // valor
  const string  descr,  // rótulo
  const uint    clr,    // color
  )
```

Parámetros

value

[in] Valor.

descr

[in] Rótulo del valor.

clr

[in] Color del valor.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ValueInsert

Inserta un nuevo valor en el diagrama (según la posición indicada).

```
bool  ValueInsert(
  const uint    pos,    // índice
  const double  value,  // valor
  const string  descr,  // rótulo
  const uint    clr,    // color
  )
```

Parámetros

pos

[in] Índice para la inserción.

value

[in] Valor.

descr

[in] Rótulo del valor.

clr

[in] Color del valor.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ValueUpdate

Actualiza los valores en el diagrama (según la posición indicada).

```
bool  ValueUpdate(
  const uint    pos,    // índice
  const double  value,  // valor
  const string  descr,  // rótulo
  const uint    clr,    // color
  )
```

Parámetros

pos

[in] El índice del valor supone el número ordinal de su adición, comenzando a partir del 0.

value

[in] Valor.

descr

[in] Rótulo del valor.

clr

[in] Color del valor.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ValueDelete

Elimina un valor del diagrama (según la posición indicada).

```
bool  ValueDelete(
  const uint  pos,  // índice
  )
```

Parámetros

pos

[in] El índice del valor supone el número ordinal de su adición, comenzando a partir del 0.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

DrawChart

Método virtual que ejecuta el dibujado del diagrama y de todos sus elementos.

```
virtual void  DrawChart()
```

© 2000-2025, MetaQuotes Ltd.

DrawPie

Dibuja un segmento del diagrama que se corresponde con un valor determinado.

```
void  DrawPie(
  double      fi3,   // ángulo del rayo que parte desde el centro del diagrama, que d
  double      fi4,   // ángulo del rayo que parte desde el centro del diagrama, que d
  int         idx,   // índice
  CPoint&     p[],  //
  const uint  clr,   //
  )
```

Parámetros

fi3

[in] Ángulo en radianes que establece el primer límite del arco.

fi4

[in] Ángulo en radianes que establece el segundo límite del arco.

idx

[in] Índice del valor al que corresponde el segmento.

p[]

[in] Matriz de los puntos de apoyo (x, y) para la construcción de los segmentos.

clr

[in] Color del segmento.

© 2000-2025, MetaQuotes Ltd.

LabelMake

Genera el rótulo de un segmento basándose en sus valores y su rótulo inicial.

```
string  LabelMake(
  const string  text,     // rótulo
  const double  value,    // valor
  const bool    to_left,  // bandera
  )
```

Parámetros

text

[in] Rótulo.

value

[in] Valor.

to_left

[in] Determina el orden de la composición del rótulo:

- true — rótulo, después el valor.
- false — valor, después el rótulo.

Valor devuelto

Rótulo del segmento.

© 2000-2025, MetaQuotes Ltd.

Gráficos 3D

En este apartado, se muestran las clases para crear gráficos en 3D construidos sobre la base de las funciones para trabajar con DirectX. La clase básica es CCnavas3D, que contiene métodos para controlar la cámara y la iluminación, y también ofrece un gestor de recursos: texturas, sombreadores, búferes de vértices, índices y parámetros de sombreadores.

Para comenzar a trabajar con la bilioteca, basta con leer el artículo Cómo crear gráficos 3D en DirectX en MetaTrader 5.

Aparte, se muestran las clases de objeto básico de escena, tales como paralelepípedos, superficies tridimensionales con datos personalizados y cuadrículas aleatorias.

© 2000-2025, MetaQuotes Ltd.

CCanvas3D

La clase CCanvas3D es una clase para la creación simplificada y la visualización de objetos en tres dimensiones en un gráfico.

Descripción

CCanvas3D simplifica significativamente la creación y visualización de grandes volúmenes de datos en forma de gráfico 3D animado. La clase contiene métodos para controlar la cámara y la iluminación, y también ofrece un gestor de recursos para crear recursos gráficos: texturas, sombreadores, búferes de vértices, índices y parámetros de sombreadores.

Además, la biblioteca contiene clases de objeto básico de escena, tales como paralelepípedos, superficies tridimensionales con datos personalizados o bien una cuadrícula aleatoria.

Para trabajar con las funciones, la tarjeta gráfica deberá soportar DX 11 y los sombreadores de la versión 5.0.

Declaración

```
class CCanvas
```

Encabezado

```
#include <Canvas\Canvas.mqh>
```

Jerarquía de herencia

CCanvas

CCanvas3D

Métodos de clase por grupos

Creación/eliminación	Descripción
Create	Crea un recurso gráfico para dibujar una escena 3D sin vinculación con el objeto del gráfico.
Attach	Obtiene del objeto OBJ_BITMAP_LABEL un recurso gráfico y lo vincula a un ejemplar de la clase CCanvas.
ObjectAdd	Añade a una escena 3D un objeto para su posterior dibujado.
Destroy	Elimina un recurso gráfico y libera un contexto gráfico 3D.
Trabajo con los colores	
AmbientColorSet	Establace el color y la intensidad de la luz ambiental circundante.
AmbientColorGet	Obtiene el color y la intensidad de la iluminación ambiental circundante.
LightDirectionSet	Establece la dirección de la fuente dirigida de luz.

© 2000-2025, MetaQuotes Ltd.

Creación/eliminación	Descripción
LightDirectionGet	Obtiene la dirección de la fuente dirigida de luz.
LightColorSet	Establace el color y la intensidad de la fuente dirigida de luz.
LightColorGet	Obtiene el color y la intensidad de la fuente dirigida de luz.
Trabajo con la cámara	
ProjectionMatrixSet	Calcula y establece la matriz de proyección de las coordenadas en 3D en un fotograma en 2D.
ProjectionMatrixGet	Obtiene la matriz de proyección de escenas 3D en un fotograma en 2D.
ViewMatrixSet	Establece la matriz de vista de la escena 3D.
ViewMatrixGet	Retorna la matriz de vista de la escena 3D.
ViewPositionSet	Establece la posición de un punto de vista hacia una escena 3D.
ViewRotationSet	Establece la dirección de la vista hacia una escena 3D.
ViewTargetSet	Establece las coordinadas del punto hacia el que está dirigida la vista.
ViewUpDirectionSet	Establece la dirección del borde superior del fotograma en el espacio 3D.
Dibujado	
Render	Ejecuta el ciclo completo de dibujado de todos los objetos de la escena en el búfer interno del fotograma para su posterior representación.
RenderBegin	Prepara el contexto gráfico para dibujar un nuevo fotograma.
RenderEnd	Copia el fotograma dibujado al búfer interno y actualiza en caso necesario la imagen el gráfico.
Obtención de recursos	
DXContext	Retorna el manejador del contexto gráfico.
DXDispatcher	Retorna el manejador del despachador de recursos.
InputScene	Retorna el puntero al búfer de parámetros de la escena.

© 2000-2025, MetaQuotes Ltd.

AmbientColorGet

Obtiene el color y la intensidad de la iluminación ambiental circundante.

```
void  AmbientColorGet(
   DXColor  &ambient_color      // color e intensidad de la iluminación ambiental
   );
```

Parámetros

&ambient_color

[out] Color de la iluminación ambiental.

Valor retornado

No

Observación

La intensidad se guarda en el canal alfa de la estructura DXColor.

© 2000-2025, MetaQuotes Ltd.

AmbientColorSet

Establace el color y la intensidad de la luz ambiental circundante.

```
void  AmbientColorSet(
   const DXColor  &ambient_color      // color e intensidad de la iluminación ambienta
   );
```

Parámetros

&ambient_color

[in] Color de la iluminación ambiental.

Valor retornado

No

Observación

La intensidad se establece en el canal alfa de la estructura DXColor.

© 2000-2025, MetaQuotes Ltd.

Attach

Obtiene del objeto OBJ_BITMAP_LABEL un recurso gráfico y lo vincula a un ejemplar de la clase CCanvas

```
bool  Attach(
   const long          chart_id,                          // identificador del grá
   const string        objname,                           // nombre del objeto
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // método de procesamient
   )
```

Crea un recurso gráfico para el objeto OBJ_BITMAP_LABEL y lo vincula a un ejemplar de la clase CCanvas.

```
bool  Attach(
   const long          chart_id,                          // identificador del grá
   const string        objname,                           // nombre del objeto
   const int           width,                             // anchura de la imagen e
   const int           height,                            // altura de la imagen en
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // método de procesamient
   )
```

Parámetros

chart_id

[in] Identificador del gráfico.

objname

[in] Denominación (nombre) del objeto gráfico.

width

[in] Anchura del fotograma en el recurso.

height

[in] Altura del fotograma.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Método de procesamiento del color. Podrá leer más información sobre el procesamiento del color en la descripción de la función ResourceCreate().

Observación

true - en caso de éxito, false - si no se ha logrado añadir el objeto gráfico.

© 2000-2025, MetaQuotes Ltd.

Create

Crea un recurso gráfico para dibujar una escena 3D sin vinculación con el objeto del gráfico.

```
virtual bool  Create(
   const string       name,                              // nombre del objeto gráfi
   const int          width,                             // anchura
   const int          height,                            // altura
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // formato del color
   );
```

Parámetros

name

[in] Nombre del objeto gráfico.

width

[in] Anchura del fotograma.

height

[in] Altura del fotograma.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Método de procesamiento del color. Podrá leer más información sobre el procesamiento del color en la descripción de la función ResourceCreate().

Observación

true si el recurso se crea con éxito, de lo contrario - false.

© 2000-2025, MetaQuotes Ltd.

Destroy

Elimina un recurso gráfico y libera un contexto gráfico 3D.

```
virtual void  Destroy()
```

Valor retornado

No

Observación

Si el recurso gráfico se ha creado con vinculación a un objeto del gráfico, el objeto del gráfico se eliminará.

© 2000-2025, MetaQuotes Ltd.

DXContext

Retorna el manejador del contexto gráfico.

```
int  DXContext()
```

Valor retornado

Manejador del contexto gráfico.

© 2000-2025, MetaQuotes Ltd.

DXDispatcher

Retorna el manejador del despachador de recursos.

```
CDXDispatcher*  DXDispatcher()
```

Valor retornado

Manejador del despachador de recursos.

© 2000-2025, MetaQuotes Ltd.

InputScene

Retorna el puntero al búfer de parámetros de la escena.

```
CDXInput*  InputScene()
```

Valor retornado

Puntero al búfer de parámetros de la escena.

© 2000-2025, MetaQuotes Ltd.

LightColorGet

Obtiene el color y la intensidad de la fuente dirigida de luz.

```
void  LightColorGet(
   DXColor  &light_color      // color e intensidad de la iluminación directa
   );
```

Parámetros

&light_color

[out] Color e intensidad de la iluminación directa.

Valor retornado

No.

Observación

La intensidad se guarda en el canal alfa de la estructura DXColor.

© 2000-2025, MetaQuotes Ltd.

LightColorSet

Establace el color y la intensidad de la fuente dirigida de luz.

```
void  LightColorSet(
   const DXColor  &light_color      // color e intensidad de la iluminación directa
   );
```

Parámetros

&light_color

[in] Color e intensidad de la iluminación directa.

Valor retornado

No.

Observación

La intensidad se establece en el canal alfa de la estructura DXColor.

© 2000-2025, MetaQuotes Ltd.

LightDirectionGet

Obtiene la dirección de la fuente dirigida de luz.

```
void  LightDirectionGet(
   DXVector3  &light_direction      // vector direccional
   );
```

Parámetros

&light_direction

[out] Vector direccional.

Valor retornado

No.

© 2000-2025, MetaQuotes Ltd.

LightDirectionSet

Establece la dirección de la fuente dirigida de luz.

```
void  LightDirectionSet(
   const DXVector3  &light_direction      // vector direccional
   );
```

Parámetros

&light_direction

[in] Vector direccional.

Valor retornado

No.

© 2000-2025, MetaQuotes Ltd.

ObjectAdd

Añade a una escena 3D un objeto para su posterior dibujado.

```
bool  ObjectAdd(
   CDXObject  *object      // puntero al objeto
   );
```

Parámetros

object

[in] Puntero al ejemplar de la clase heredera de la clase abstracta CDXObject.

Valor retornado

true - en caso de éxito, false - si no se ha logrado añadir el objeto gráfico 3D.

© 2000-2025, MetaQuotes Ltd.

ProjectionMatrixGet

Obtiene la matriz de proyección de escenas 3D en un fotograma en 2D.

```
void  ProjectionMatrixGet(
   DXMatrix  &projection_matrix      // matriz de proyección
   );
```

Parámetros

&projection_matrix

[out] Matriz de proyección.

Valor retornado

No.

© 2000-2025, MetaQuotes Ltd.

ProjectionMatrixSet

Calcula y establece la matriz de proyección de las coordenadas en 3D en un fotograma en 2D.

```
void  ProjectionMatrixSet(
   float  fov,              // ángulo de visión (field of view)
   float  aspect_ratio,     // relación de los lados del fotograma
   float  z_near,           //
   float  z_far             //
   );
```

Parámetros

fov

[in] Anchura del campo visual en radianes para crear la proyección de la escena.

aspect_ratio

[in] Relación de los lados del fotograma 2D.

z_near

[in] Distancia hasta el plano de recorte próximo.

z_far

[in] Distancia hasta el plano de recorte lejano.

Valor retornado

No.

Observación

En el fotograma en 2D solo se mostrarán las proyecciones de aquellos objetos 3D que entren en el campo visual indicado y se ubiquen entre los planos de recorte próximo y lejano.

© 2000-2025, MetaQuotes Ltd.

Render

Ejecuta el ciclo completo de dibujado de todos los objetos de la escena en el búfer interno del fotograma para su posterior representación.

```
bool  Render(
   uint  flags,                  // combinación de banderas
   uint  background_color=0      // color del fondo
   );
```

Parámetros

flags

[in] Combinación de banderas que establece el modo de dibujado. Posibles valores:
DX_CLEAR_COLOR - limpiar el búfer de imagen con el color *background_color*.
DX_CLEAR_DEPTH - limpiar el búfer de profundidad.

background_color=0

[in] Color de fondo del fotograma 3D.

Valor retornado

true - en caso de éxito, false - si no se ha logrado realizar el dibujado.

Observación

La llamada de Render() no provoca la actualización de la escena en el gráfico, solo actualiza el búfer interno de la imagen. Para dibujar el fotograma actualizado, deberemos llamar explícitamente al método Update().

Render() contiene las llamadas a RenderBegin y RenderEnd().

© 2000-2025, MetaQuotes Ltd.

RenderBegin

Prepara el contexto gráfico para dibujar un nuevo fotograma.

```
virtual bool  RenderBegin(
   uint  flags,                  // combinación de banderas
   uint  background_color=0      // color del fondo
   );
```

Parámetros

flags

[in] Combinación de banderas que establece el modo de dibujado. Posibles valores:
DX_CLEAR_COLOR - limpiar el búfer de imagen con el color *background_color*.
DX_CLEAR_DEPTH - limpiar el búfer de profundidad.

background_color=0

[in] Color de fondo del fotograma 3D.

Valor retornado

true - en caso de éxito, false - si no se ha logrado actualizar los parámetros de entrada del sombreador.

© 2000-2025, MetaQuotes Ltd.

RenderEnd

Copia el fotograma dibujado al búfer interno y actualiza en caso necesario la imagen el gráfico.

```
virtual bool  RenderEnd(
   bool  redraw=false      // bandera de actualización
   );
```

Parámetros

redraw=false

[in] Bandera que indica la necesidad de redibujar el gráfico.

Valor retornado

true en el caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ViewMatrixGet

Retorna la matriz de vista de la escena 3D.

```
void  ViewMatrixGet(
   DXMatrix  &view_matrix      // matriz de vista
   );
```

Parámetros

&view_matrix

[out] Matriz de vista que establece la posición y la dirección de la cámara en un espacio 3D.

Valor retornado

No.

© 2000-2025, MetaQuotes Ltd.

ViewMatrixSet

Establece la matriz de vista de la escena 3D.

```
void  ViewMatrixSet(
   const DXMatrix  &view_matrix      // matriz de vista
   );
```

Parámetros

&view_matrix

[in] Matriz de vista que establece la posición y la dirección de la cámara en un espacio 3D.

Valor retornado

No.

© 2000-2025, MetaQuotes Ltd.

ViewPositionSet

Establece la posición de un punto de vista hacia una escena 3D.

```
void  ViewPositionSet(
   const DXVector3  &position      // posición del punto de vista
   );
```

Parámetros

&posiiton

[in] Posición del punto de vista hacia una escena 3D.

Valor retornado

No.

Observación

El establecimiento de la posición del punto de vista con la ayuda de ViewPositionSet() cambia la matriz de vista obtenida en ViewMatrixGet().

© 2000-2025, MetaQuotes Ltd.

ViewRotationSet

Establece la dirección de la vista hacia una escena 3D.

```
void  ViewRotationSet(
   const DXVector3  &rotation      // vector del ángulo de giro
   );
```

Parámetros

&rotation

[in] Vector que establece el ángulo de Euler para calcular la dirección de la mirada hacia una escena 3D.

Valor retornado

No.

Observación

El establecimiento de la dirección de la mirada con la ayuda de ViewRotationSet() cambia la matriz de vista obtenida en ViewMatrixGet().

© 2000-2025, MetaQuotes Ltd.

ViewTargetSet

Establece las coordinadas del punto hacia el que está dirigida la vista.

```
void  ViewTargetSet(
   const DXVector3  &target      // coordenadas del objetivo
   );
```

Parámetros

&target

[in] Coordenadas del punto al que está dirigida la mirada.

Valor retornado

No.

Observación

Se usa para fijar la mirada en un punto de la escena al desplazar el punto de vista.

El establecimiento de una nueva coordenada de objetivo con la ayuda de ViewRotationSet() cambia la matriz de vista obtenida en ViewMatrixGet().

ViewTargetSet() se usa junto con ViewUpDirectionSet() para determinar la dirección de la mirada.

© 2000-2025, MetaQuotes Ltd.

ViewUpDirectionSet

Establece la dirección del borde superior del fotograma en el espacio 3D.

```
void  ViewUpDirectionSet(
   const DXVector3  &up_direction      // dirección superior
   );
```

Parámetros

&up_direction

[in] Dirección superior del fotograma en el espacio 3D.

Valor retornado

No.

Observación

El establecimiento de una nueva dirección con la ayuda de ViewUpDirectionSet() cambia la matriz de vista obtenida en ViewMatrixGet().

ViewUpDirectionSet() se usa junto con ViewTargetSet() para determinar la dirección de la mirada.

© 2000-2025, MetaQuotes Ltd.

CChart

La clase CChart facilita el acceso a las propiedades del objeto gráfico "Chart".

Descripción

La clase CChart proporciona acceso a las propiedades del objeto "Chart".

Declaración

```
class CChart : public CObject
```

Título

```
#include <Charts\Chart.mqh>
```

Jerarquía de herencia

CObject

CChart

Métodos de la clase

Acceso a datos protegidos	
ChartID	Obtiene el identificador del gráfico
Propiedades generales	
Mode	Obtiene/Establece el valor de la propiedad "Mode" (modo: barra, vela o línea)
Foreground	Obtiene/Establece el valor de la propiedad "Foreground" (primer plano)
Shift	Obtiene/Establece el valor de la propiedad "Shift (desplazamiento)
ShiftSize	Obtiene/Establece el valor de la propiedad "ShiftSize" (tamaño del desplazamiento, en porcentaje)
AutoScroll	Obtiene/Establece el valor de la propiedad "AutoScroll"
Scale	Obtiene/Establece el valor de la propiedad "Scale" (escala)
ScaleFix	Obtiene/Establece el valor de la propiedad "ScaleFix" (indica si la escala del gráfico es fija)
ScaleFix_11	Obtiene/Establece el valor de la propiedad "ScaleFix_11" (la escala de gráfico es 1:1, o no)
FixedMax	Obtiene/Establece el valor de la propiedad "FixedMax" (precio máximo fijo)
FixedMin	Obtiene/Establece el valor de la propiedad "FixedMin" (precio mínimo fijo)

© 2000-2025, MetaQuotes Ltd.

Acceso a datos protegidos	
ScalePPB	Obtiene/Establece el valor de la propiedad "ScalePPB" (la escala es "punto por barra")
PointsPerBar	Obtiene/Establece el valor de la propiedad "PointsPerBar" (en puntos por barra)
Mostrar las propiedades	
ShowOHLC	Obtiene/Establece el valor de la propiedad "ShowOHLC"
ShowLineBid	Obtiene/Establece el valor de la propiedad "ShowLineBid"
ShowLineAsk	Obtiene/Establece el valor de la propiedad "ShowLineAsk"
ShowLastLine	Obtiene/Establece el valor de la propiedad "ShowLastLine"
ShowPeriodSep	Obtiene/Establece el valor de la propiedad "ShowPeriodSep" (muestra los separadores del periodo)
ShowGrid	Obtiene/Establece el valor de la propiedad "ShowGrid"
ShowVolumes	Obtiene/Establece el valor de la propiedad "ShowVolumes" (color de los volúmenes y niveles de las posiciones abiertas)
ShowObjectDescr	Obtiene/Establece el valor de la propiedad "ShowObjectDescr" (muestra la descripción de los objetos gráficos)
ShowDateScale	Establece el valor de la propiedad "ShowDateScale" (escala de la fecha del gráfico)
ShowPriceScale	Establece el valor de la propiedad "ShowPriceScale" (escala del precio del gráfico)
Propiedades de los colores	
ColorBackground	Obtiene/Establece el valor de la propiedad "ColorBackground" (color de fondo del gráfico)
ColorForeground	Obtiene/Establece el valor de la propiedad "ColorForeground" (color de los ejes, escala y strings OHLC del gráfico)
ColorGrid	Obtiene/Establece el valor de la propiedad "ColorGrid" (color de la cuadrícula)
ColorBarUp	Obtiene/Establece el valor de la propiedad "ColorBarUp" (color de las barras alcistas, su sombra y los contornos del cuerpo de la vela)
ColorBarDown	Obtiene/Establece el valor de la propiedad "ColorBarDown" (color de las barras bajistas, su sombra y los contornos del cuerpo de la vela)

© 2000-2025, MetaQuotes Ltd.

Acceso a datos protegidos	
ColorCandleBull	Obtiene/Establece el valor de la propiedad "ColorCandleBull" (color del cuerpo de la vela alcista)
ColorCandleBear	Obtiene/Establece el valor de la propiedad "ColorCandleBear" (color del cuerpo de la vela bajista)
ColorChartLine	Obtiene/Establece el valor de la propiedad "ColorChartLine" (color de la línea del gráfico y velas Doji)
ColorVolumes	Obtiene/Establece el valor de la propiedad "ColorVolumes" (color de los volúmenes y niveles de las posiciones abiertas)
ColorLineBid	Obtiene/Establece el valor de la propiedad "ColorLineBid" (color de la línea Bid)
ColorLineAsk	Obtiene/Establece el valor de la propiedad "ColorLineAsk" (color de la línea Ask)
ColorLineLast	Obtiene/Establece el valor de la propiedad "ColorLineLast" (color de la línea del precio de la última transacción)
ColorStopLevels	Obtiene/Establece el valor de la propiedad "ColorStopLevels" (color de los niveles SL y TP)
Propiedades de solo lectura	
VisibleBars	Obtiene el número total de barras visibles del gráfico
WindowsTotal	Obtiene el número total de ventanas de gráfico, incluyendo las subventanas del indicador
WindowIsVisible	Obtiene la bandera de visibilidad de la subventana del gráfico especificada
WindowHandle	Obtiene el manejador de la ventana del gráfico (HWND)
FirstVisibleBar	Obtiene el número de la primera barra visible del gráfico
WidthInBars	Obtiene la anchura de la ventana en barras.
WidthInPixels	Obtiene la anchura de la subventana en píxeles.
HeightInPixels	Obtiene la altura de la subventana en píxeles.
PriceMin	Obtiene el precio mínimo de la subventana especificada
PriceMax	Obtiene el precio máximo de la subventana especificada
Propiedades	
Attach	Asigna el gráfico actual a la instancia de la clase
FirstChart	Asigna el primer gráfico del terminal cliente a la instancia de la clase

© 2000-2025, MetaQuotes Ltd.

Acceso a datos protegidos	
NextChart	Asigna el gráfico siguiente del terminal cliente a la instancia de la clase
Open	Abre el gráfico con los parámetros especificados y lo asigna a la instancia de la clase
Detach	Desvincula el gráfico de la instancia de la clase
Close	Cierra el gráfico, asignado a la instancia de la clase
BringToTop	Muestra el gráfico encima de otros gráficos
EventObjectCreate	Establece la bandera para enviar notificaciones de eventos de creación de objetos en un gráfico
EventObjectDelete	Establece la bandera para enviar notificaciones de eventos de borrado de objetos en un gráfico
Indicadores	
IndicatorAdd	Añade el indicador con el manejador dado en la subventana del gráfico especificada
IndicatorDelete	Borra el indicador con el nombre dado de la subventana del gráfico especificada
IndicatorsTotal	Devuelve el nombre de todos los indicadores aplicados en la subventana del gráfico especificada
IndicatorName	Devuelve el nombre corto del indicador en la subventana del gráfico especificada
Navegación	
Navigate	Navega por el gráfico
Acceso al API de MQL5	
Symbol	Obtiene el símbolo del gráfico
Period	Obtiene el periodo del gráfico
Redraw	Redibuja el gráfico, asignado a la instancia de la clase
GetInteger	La función devuelve el valor de la propiedad de objeto correspondiente
SetInteger	Establece el nuevo valor de la propiedad de tipo integer
GetDouble	La función devuelve el valor de la propiedad de objeto correspondiente
SetDouble	Establece el nuevo valor de la propiedad de tipo double
GetString	La función devuelve el valor de la propiedad de objeto correspondiente

© 2000-2025, MetaQuotes Ltd.

Acceso a datos protegidos	
SetString	Establece el nuevo valor de la propiedad de tipo string
SetSymbolPeriod	Cambia el símbolo y el periodo del gráfico, asignado a la instancia de la clase
ApplyTemplate	Aplica la plantilla especificada al gráfico
ScreenShot	Crea una captura de pantalla del gráfico especificado y la guarda en un archivo .gif
WindowOnDropped	Obtiene la subventana del gráfico correspondiente al punto de soltura del objeto (experto o script)
PriceOnDropped	Obtiene la coordenada del precio correspondiente al punto de soltura del objeto (experto o script)
TimeOnDropped	Obtiene la coordenada temporal correspondiente al punto de soltura del objeto (experto o script)
XOnDropped	Obtiene la coordenada X correspondiente al punto de soltura del objeto (experto o script)
YOnDropped	Obtiene la coordenada Y correspondiente al punto de soltura del objeto (experto o script)
Entrada/Salida	
virtual Save	Guarda los parámetros del objeto en el archivo
virtual Load	Carga los parámetros del objeto a partir del archivo
virtual Type	Obtiene el identificador de tipo de objeto del gráfico

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

© 2000-2025, MetaQuotes Ltd.

ChartID

Devuelve el identificador del gráfico.

```
long  ChartID() const
```

Valor devuelto

Identificador del gráfico, asignado a la instancia de la clase. Si no hay ningún objeto asignado devuelve -1.

© 2000-2025, MetaQuotes Ltd.

Mode (Método Get)

Obtiene el valor de la propiedad "Mode" (barra, vela o línea).

```
ENUM_CHART_MODE  Mode() const
```

Valor devuelto

Valor de la propiedad "Mode" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve WRONG_VALUE.

Mode (Método Set)

Establece un nuevo valor para la propiedad "Mode" (barra, vela o línea).

```
bool  Mode(
   ENUM_CHART_MODE  mode       // nuevo modo gráfico
   )
```

Parámetros

mode

[in] Modo gráfico (vela, barra o línea) de la enumeración ENUM_CHART_MODE.

Valor devuelto

true si se ejecuta correctamente, false si el modo no ha sido cambiado.

© 2000-2025, MetaQuotes Ltd.

Foreground (Método Get)

Obtiene el valor de la propiedad "Foreground".

```
bool  Foreground() const
```

Valor devuelto

Valor de la propiedad "Foreground" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

Foreground (Método Set)

Establece el nuevo valor de la propiedad "Foreground".

```
bool  Foreground(
   bool  foreground      // nuevo valor de la bandera
   )
```

Parámetros

foreground

[in] Nuevo valor de la propiedad "Foreground".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

Shift (Método Get)

Obtiene el valor de la propiedad "Shift".

```
bool  Shift() const
```

Valor devuelto

Valor de la propiedad "Shift" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

Shift (Método Set)

Establece un nuevo valor para la propiedad "Shift".

```
bool  Shift(
   bool  shift      // nuevo valor de la ventana
   )
```

Parámetros

shift

[in] Nuevo valor de la propiedad "Shift".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShiftSize (Método Get)

Obtiene el valor de la propiedad "ShiftSize" (en porcentaje).

```
double  ShiftSize() const
```

Valor devuelto

Valor de la propiedad "ShiftSize" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

ShiftSize (Método Set)

Establece el nuevo valor de la propiedad "Shift" (en porcentaje).

```
bool  ShiftSize(
   double  shift_size      // nuevo valor de la propiedad
   )
```

Parámetros

shift_size

[in] Nuevo valor de la propiedad "ShiftSize" (en porcentaje).

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

AutoScroll (Método Get)

Obtiene el valor de la propiedad "AutoScroll".

```
bool  AutoScroll() const
```

Valor devuelto

Valor de la propiedad "AutoScroll" del objeto, asignado a la instancia de clase. Si no hay ningún objeto asignado, devuelve false.

AutoScroll (Método Set)

Establece un nuevo valor a la propiedad "AutoScroll".

```
bool  AutoScroll(
   bool  autoscroll      // nuevo valor de la bandera
   )
```

Parámetros

autoscroll

[in] Nuevo valor de la propiedad "Autoscroll".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

Scale (Método Get)

Obtiene el valor de la propiedad "Scale".

```
int  Scale() const
```

Valor devuelto

Valor de la propiedad "Scale" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve 0.

Scale (Método Set)

Establece un nuevo valor para la propiedad "Scale".

```
bool  Scale(
   int  scale      // valor nuevo
   )
```

Parámetros

scale

[in] Nuevo valor de la propiedad "Scale".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ScaleFix (Método Get)

Obtiene el valor de la propiedad "ScaleFix".

```
bool  ScaleFix() const
```

Valor devuelto

Valor de la propiedad "ScaleFix" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

ScaleFix (Método Set)

Establece el nuevo valor de la propiedad "ScaleFix".

```
bool  ScaleFix(
   bool  scale_fix      // valor nuevo
   )
```

Parámetros

scale_fix

[in] Nuevo valor de la propiedad "ScaleFix".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ScaleFix_11 (Método Get)

Obtiene el valor de la propiedad "ScaleFix_11".

```
bool  ScaleFix_11() const
```

Valor devuelto

Valor de la propiedad "ScaleFix_11" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

ScaleFix_11 (Método Set)

Establece el nuevo valor de la propiedad "ScaleFix_11".

```
bool  ScaleFix_11(
   string  scale_11      // valor nuevo
   )
```

Parámetros

scale_11

[in] Nuevo valor de la propiedad "ScaleFix_11".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

FixedMax (Método Get)

Obtiene el valor de la propiedad "FixedMax" (precio máximo fijo).

```
double  FixedMax() const
```

Valor devuelto

Valor de la propiedad "FixedMax" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

FixedMax (Método Set)

Establece un nuevo valor para la propiedad "FixedMax".

```
bool  FixedMax(
   double  max       // nuevo máximo fijo
   )
```

Parámetros

max

[in] Nuevo valor de la propiedad "FixedMax".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

FixedMin (Método Get)

Obtiene el valor de la propiedad "FixedMin" (precio mínimo fijo).

```
double  FixedMin() const
```

Valor devuelto

Valor de la propiedad "FixedMin" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

FixedMin (Método Set)

Establece el nuevo valor de la propiedad "FixedMin".

```
bool  FixedMax(
   double  min      // nuevo mínimo fijo
   )
```

Parámetros

max

[in] Nuevo valor de la propiedad "FixedMin".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

PointsPerBar (Método Get)

Obtiene el valor de la propiedad "PointsPerBar" (en puntos por barra).

```
double  PointsPerBar() const
```

Valor devuelto

Valor de la propiedad "PointsPerBar" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve EMPTY_VALUE.

PointsPerBar (Método Set)

Establece un nuevo valor para la propiedad "PointsPerBar".

```
bool  PointsPerBar(
   double  ppb      // nueva escala (en puntos por barra)
   )
```

Parámetros

ppb

[in] Nuevo valor de la escala (en puntos por barra).

Valor devuelto

true si se ejecuta correctamente, false si la escala no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ScalePPB (Método Get)

Obtiene el valor de la propiedad "ScalePPB".

```
bool  ScalePPB() const
```

Valor devuelto

Valor de la propiedad "ScalePPB" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

ScalePPB (Método Set)

Establece el nuevo valor de la propiedad "ScalePPB".

```
bool  ScalePPB(
   bool  scale_ppb      // nuevo valor de la bandera
   )
```

Parámetros

scale_ppb

[in] Nuevo valor de la propiedad "ScalePPB".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShowOHLC (Método Get)

Obtiene el valor de la propiedad "ShowOHLC".

```
bool  ShowOHLC() const
```

Valor devuelto

Valor de la propiedad "ShowOHLC" del objeto, asignado a la instancia de la clase. Si no hay ningún objeto asignado, devuelve false.

ShowOHLC (Método Set)

Establece el nuevo valor de la propiedad "ShowOHLC".

```
bool  ShowOHLC(
   bool  show      // valor nuevo
   )
```

Parámetros

show

[in] Nuevo valor de la propiedad "ShowOHLC".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShowLineBid (Método Get)

Obtiene el valor de la propiedad "ShowLineBid".

```
bool  ShowLineBid() const
```

Valor devuelto

Valor de la propiedad "ShowLineBid" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve false.

ShowLineBid (Método Set)

Establece el nuevo valor de la propiedad "ShowLineBid".

```
bool  ShowLineBid(
   bool  show      // valor nuevo
   )
```

Parámetros

show

[in] Nuevo valor de la propiedad "ShowLineBid".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShowLineAsk (Método Get)

Obtiene el valor de la propiedad "ShowLineAsk".

```
bool  ShowLineAsk() const
```

Valor devuelto

Valor de la propiedad "ShowLineAsk" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve false.

ShowLineAsk (Método Set)

Establece el nuevo valor de la propiedad "ShowLineAsk".

```
bool  ShowLineAsk(
   bool  show      // valor nuevo
   )
```

Parámetros

show

[in] Nuevo valor de la propiedad "ShowLineAsk".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShowLastLine (Método Get)

Obtiene el valor de la propiedad "ShowLastLine".

```
bool  ShowLastLine() const
```

Valor devuelto

Valor de la propiedad "ShowLastLine" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve false.

ShowLastLine (Método Set)

Establece el nuevo valor de la propiedad "ShowLastLine".

```
bool  ShowLastLine(
   bool  show      // nuevo valor de la bandera
   )
```

Parámetros

show

[in] Nuevo valor de la propiedad "ShowLastLine".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShowPeriodSep (Método Get)

Obtiene el valor de la propiedad "ShowPeriodSep" (muestra los separadores de los períodos temporales).

```
bool  ShowPeriodSep() const
```

Valor devuelto

Valor de la propiedad "ShowPeriodSep" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve false.

ShowPeriodSep (Método Set)

Establece un nuevo valor para la propiedad "ShowPeriodSep".

```
bool  ShowPeriodSep(
   bool  show      // valor nuevo
   )
```

Parámetros

show

[in] Nuevo valor de la propiedad "ShowPeriodSep".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShowGrid (Método Get)

Obtiene el valor de la propiedad "ShowGrid".

```
bool  ShowGrid() const
```

Valor devuelto

Valor de la propiedad "ShowGrid" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve false.

ShowGrid (Método Set)

Establece el nuevo valor de la propiedad "ShowGrid".

```
bool  ShowGrid(
   bool  show      // valor nuevo
   )
```

Parámetros

show

[in] Nuevo valor de la propiedad "ShowGrid".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShowVolumes (Método Get)

Obtiene el valor de la propiedad "ShowVolumes".

```
bool  ShowVolumes() const
```

Valor devuelto

Valor de la propiedad "ShowVolumes" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve false.

ShowVolumes (Método Set)

Establece un valor nuevo para la propiedad "ShowVolumes".

```
bool  ShowVolumes(
   bool  show      // valor nuevo
   )
```

Parámetros

show

[in] Nuevo valor de la propiedad "ShowVolumes".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShowObjectDescr (Método Get)

Obtiene el valor de la propiedad "ShowObjectDescr" (muestra la descripción del objeto gráfico).

```
bool  ShowObjectDescr() const
```

Valor devuelto

Valor de la propiedad "ShowObjectDescr" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve false.

ShowObjectDescr (Método Set)

Establece el nuevo valor de la propiedad "ShowObjectDescr".

```
bool  ShowObjectDescr(
   bool  show      // Nuevo valor
   )
```

Parámetros

show

[in] Nuevo valor de la propiedad "ShowObjectDescr".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShowDateScale

Establece el nuevo valor de la propiedad "ShowDateScale".

```
bool  ShowDateScale(
   bool  show      // Nuevo valor
   )
```

Parámetros

show

[in] Nuevo valor de la propiedad "ShowDateScale".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ShowPriceScale

Establece el nuevo valor de la propiedad "ShowPriceScale".

```
bool  ShowPriceScale(
   bool  show      // Nuevo valor
   )
```

Parámetros

show

[in] Nuevo valor de la propiedad "ShowPriceScale".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ColorBackground (Método Get)

Obtiene el valor de la propiedad "ColorBackground" (color de fondo del gráfico).

```
color  ColorBackground() const
```

Valor devuelto

Valor de la propiedad "ColorBackground" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorBackground (Método Set)

Establece un nuevo valor para la propiedad "ColorBackground".

```
bool  ColorBackground(
   color  new_color      // nuevo color de fondo
   )
```

Parámetros

new_color

[in] Nuevo color de fondo.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorForeground (Método Get)

Obtiene el valor de la propiedad "ColorForeground" (color de los ejes, escala y cadenas OHLC del gráfico).

```
color  ColorForeground() const
```

Valor devuelto

Valor de la propiedad "ColorForeground" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorForeground (Método Set)

Establece un nuevo valor para la propiedad "ColorForeground" (ejes, escala y cadena OHLC).

```
bool  ColorForeground(
   color  new_color      // Color nuevo
   )
```

Parámetros

new_color

[in] Nuevo color de los ejes, escala y cadena OHLC.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorGrid (Método Get)

Establece el valor de la propiedad "ColorGrid" (color de la cuadrícula).

```
color  ColorGrid() const
```

Valor devuelto

Valor de la propiedad "ColorGrid" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorGrid (Método Set)

Establece un valor nuevo para la propiedad "ColorGrid".

```
bool  ColorGrid(
   color  new_color      // nuevo color de la cuadrícula
   )
```

Parámetros

new_color

[in] Nuevo color de la cuadrícula.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorBarUp (Método Get)

Obtiene el valor de la propiedad "ColorBarUp" (color de las barras alcistas, sombra y contorno del cuerpo de la vela).

```
color  ColorBarUp() const
```

Valor devuelto

Valor de la propiedad "ColorBarUp" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorBarUp (Método Set)

Establece un valor nuevo para la propiedad "ColorBarUp".

```
bool  ColorBarUp(
   color  new_color      // nuevo color de las barras alcistas
   )
```

Parámetros

new_color

[in] Nuevo color de las barras alcistas.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorBarDown (Método Get)

Obtiene el valor de la propiedad "ColorBarDown" (color de las barras bajistas, sombra y contorno del cuerpo de la vela).

```
color  ColorBarDown() const
```

Valor devuelto

Valor de la propiedad "ColorBarDown" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorBarDown (Método Set)

Establece un nuevo valor para la propiedad "ColorBarDown".

```
bool  ColorBarDown(
   color  new_color      // nuevo color de las barras bajistas
   )
```

Parámetros

new_color

[in] Nuevo color de las barras.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorCandleBull (Método Get)

Obtiene el valor de la propiedad "ColorCandleBull" (color del cuerpo de la vela alcista).

```
color  ColorCandleBull() const
```

Valor devuelto

Valor de la propiedad "ColorCandleBull" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorCandleBull (Método Set)

Establece un nuevo valor para la propiedad "ColorBarBull".

```
bool  ColorCandleBull(
   color  new_color      // color nuevo del cuerpo de la vela alcista
   )
```

Parámetros

new_color

[in] Color nuevo del cuerpo de la vela alcista.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorCandleBear (Método Get)

Obtiene el valor de la propiedad "ColorCandleBear" (color del cuerpo de la vela bajista).

```
color  ColorCandleBear() const
```

Valor devuelto

Valor de la propiedad "ColorCandleBear" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorCandleBear (Método Set)

Establece un nuevo valor para la propiedad "ColorBarBear".

```
bool  ColorCandleBear(
   color  new_color      // nuevo color del cuerpo de la vela bajista
   )
```

Parámetros

new_color

[in] Nuevo color del cuerpo de la vela bajista.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorChartLine (Método Get)

Obtiene el valor de la propiedad "ColorChartLine" (color de la línea del gráfico y velas Doji).

```
color  ColorChartLine() const
```

Valor devuelto

Valor de la propiedad "ColorChartLine" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorChartLine (Método Set)

Establece un valor nuevo para la propiedad "ColorChartLine".

```
bool  ColorChartLine(
   color  new_color      // color nuevo de las líneas del gráfico
   )
```

Parámetros

new_color

[in] Color nuevo de las líneas del gráfico (velas Doji).

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorVolumes (Método Get)

Obtiene el valor de la propiedad "ColorVolumes" (color de los volúmenes y de los niveles de las posiciones abiertas).

```
color  ColorVolumes() const
```

Valor devuelto

Valor de la propiedad "ColorVolumes" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorVolumes (Método Set)

Establece un nuevo valor para la propiedad "ColorVolumes".

```
bool  ColorVolumes(
   color  new_color      // nuevo color de los volúmenes (niveles de las posiciones ab
   )
```

Parámetros

new_color

[in] Nuevo color de los volúmenes (niveles de las posiciones abiertas).

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorLineBid (Método Get)

Obtiene el valor de la propiedad "ColorLineBid" (color de la línea Bid).

```
color  ColorLineBid() const
```

Valor devuelto

Valor de la propiedad "ColorLineBid" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorLineBid (Método Set)

Establece un nuevo valor para la propiedad "ColorLineBid".

```
bool  ColorLineBid(
   color  new_color      // nuevo color de la línea Bid
   )
```

Parámetros

new_color

[in] Nuevo color de la línea Bid.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorLineAsk (Método Get)

Obtiene el valor de la propiedad "ColorLineAsk" (color de la línea Ask).

```
color  ColorLineAsk() const
```

Valor devuelto

Valor de la propiedad "ColorLineAsk" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorLineAsk (Método Set)

Establece el nuevo valor de la propiedad "ColorLineAsk".

```
bool  ColorLineAsk(
   color  new_color      // nuevo color de la línea Ask
   )
```

Parámetros

new_color

[in] Nuevo color de la línea Ask.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorLineLast (Método Get)

Obtiene el valor de la propiedad "ColorLineLast" (color de la última línea de precio).

```
color  ColorLineLast() const
```

Valor devuelto

Valor de la propiedad "ColorLineLast" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorLineLast (Método Set)

Establece el nuevo valor de la propiedad "ColorLineLast".

```
bool  ColorLineLast(
   color  new_color      // color nuevo de la última línea del precio
   )
```

Parámetros

new_color

[in] Color nuevo de la última línea del precio.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

ColorStopLevels (Método Get)

Obtiene el valor de la propiedad "ColorStopLevels" (color de los niveles SL y TP).

```
color  ColorStopLevels() const
```

Valor devuelto

Valor de la propiedad "ColorStopLevels" del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve CLR_NONE.

ColorStopLevels (Método Set)

Establece el nuevo valor de la propiedad "ColorStopLevels".

```
bool  ColorStopLevels(
   color  new_color      // nuevo color de los niveles de precio SL y TP
   )
```

Parámetros

new_color

[in] Nuevo color de los niveles de precio Stop Loss y Take Profit.

Valor devuelto

true si se ejecuta correctamente, false si el color no se ha podido cambiar.

© 2000-2025, MetaQuotes Ltd.

VisibleBars

Obtiene el número total de barras visibles del gráfico.

```
int  VisibleBars() const
```

Valor devuelto

Obtiene el número total de barras visibles del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

WindowsTotal

Obtiene el número total de ventanas de gráfico, incluyendo las subventanas de los gráficos de los indicadores.

```
int  WindowsTotal() const
```

Valor devuelto

Número total de ventanas, incluyendo las subventanas de los gráficos de los indicadores, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

WindowIsVisible

Obtiene la bandera de visibilidad de la subventana del gráfico especificada.

```
bool  WindowIsVisible(
   int  num      // número de subventana
   ) const
```

Parámetros

num

[in] Número de subventana (0 significa ventana base).

Valor devuelto

Devuelve la bandera de visibilidad de la subventana del gráfico especificada, asignada a la instancia del gráfico. Si no hay ningún gráfico asignado, devuelve false.

© 2000-2025, MetaQuotes Ltd.

WindowHandle

Obtiene el manejador de ventana del gráfico (HWND).

```
int  WindowHandle() const
```

Valor devuelto

Manejador de ventana del gráfico, asignado a la instancia del gráfico. Si no hay ningún gráfico asignado, devuelve INVALID_HANDLE.

© 2000-2025, MetaQuotes Ltd.

FirstVisibleBar

Obtiene el número de la primera barra visible del gráfico.

```
int  FirstVisibleBar() const
```

Valor devuelto

Número de la primera barra visible del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve -1.

© 2000-2025, MetaQuotes Ltd.

WidthInBars

Obtiene la anchura de la ventana en barras.

```
int  WidthInBars() const
```

Valor devuelto

Anchura de la ventana en barras, asignada a la instancia del gráfico. Si no hay ningún gráfico asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

WidthInPixels

Obtiene la anchura de la subventana en píxeles.

```
int  WidthInPixels() const
```

Valor devuelto

Anchura de la subventana en píxeles, asignada a la instancia del gráfico. Si no hay ningún gráfico asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

HeightInPixels

Obtiene la altura de la subventana en píxeles.

```
int  HeightInPixels(
   int  num      // número de subventana
   ) const
```

Parámetros

num

[in] Número de subventana (0 significa ventana base).

Valor devuelto

Altura de la subventana en píxeles, asignada a la instancia del gráfico. Si no hay ningún gráfico asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

PriceMin

Obtiene el precio mínimo de la subventana especificada.

```
double  PriceMin(
   int  num      // número de subventana
   ) const
```

Parámetros

num

[in] Número de subventana (0 significa ventana base).

Valor devuelto

Valor del precio mínimo del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

PriceMax

Obtiene el precio máximo de la subventana especificada.

```
double  PriceMax(
   int  num      // número de subventana
   ) const
```

Parámetros

num

[in] Número de subventana (0 significa ventana base).

Valor devuelto

Valor del precio máximo del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Attach

Asigna el gráfico actual a la instancia de la clase.

```
void  Attach()
```

Attach

Asigna el gráfico especificado a la instancia de la clase.

```
void  Attach(
   long  chart       // Identificador del gráfico
   )
```

Parámetros

chart

[in] Identificador del gráfico a asignar.

© 2000-2025, MetaQuotes Ltd.

FirstChart

Asigna a la instancia de la clase el primer gráfico del terminal cliente.

```
void  FirstChart()
```

© 2000-2025, MetaQuotes Ltd.

NextChart

Asigna el próximo gráfico del terminal cliente a la instancia de la clase.

```
void  NextChart()
```

© 2000-2025, MetaQuotes Ltd.

Open

Abre el gráfico con los parámetros especificados y lo asigna a la instancia de la clase.

```
long  Open(
   const string     symbol_name,     // Nombre del símbolo
   ENUM_TIMEFRAMES  timeframe        // Período
   )
```

Parámetros

symbol_name

[in] Nombre del símbolo. NULL significa el símbolo del gráfico actual (donde se adjunta el experto).

timeframe

[in] Período de tiempo del gráfico (enumeración ENUM_TIMEFRAMES). 0 significa el período de tiempo actual.

Valor devuelto

Identificador del gráfico.

© 2000-2025, MetaQuotes Ltd.

Detach

Desvincula el gráfico de la instancia de la clase.

```
void  Detach()
```

© 2000-2025, MetaQuotes Ltd.

Close

Cierra el gráfico, asignado a la instancia de la clase.

```
void  Close()
```

© 2000-2025, MetaQuotes Ltd.

BringToTop

Muestra el gráfico encima de los demás gráficos.

```
bool  BringToTop() const
```

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

EventObjectCreate

Establece una bandera para enviar notificaciones de un evento de creación de nuevo objeto a todos los programas MQL5 de un gráfico.

```
bool  EventObjectCreate(
   bool  flag      // bandera
   )
```

Parámetros

flag

[in] Nuevo valor de la bandera.

Valor devuelto

true - si se ejecuta correctamente, false - si la bandera no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

EventObjectDelete

Establece una bandera para enviar notificaciones de un evento de borrado de objeto a todos los programas MQL5 de un gráfico.

```
bool  EventObjectDelete(
   bool  flag      // bandera
   )
```

Parámetros

flag

[in] Nuevo valor de la bandera.

Valor devuelto

true - si se ejecuta correctamente, false - si la bandera no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

IndicatorAdd

Añade a la ventana de gráfico dada un indicador con el manejador especificado.

```
bool  IndicatorAdd(
   int   sub_win          // número de subventana
   int   handle           // manejador del indicador
   );
```

Parámetros

sub_win

[in] Número de subventana del gráfico. 0 significa la ventana de gráfico principal. si se especifica un número de ventana que no existe, se creará una nueva ventana.

handle

[in] El manejador del indicador.

Valor devuelto

La función devuelve true si se ejecuta correctamente, en otro caso devuelve false. Para obtener información sobre el error, llamar a la función GetLastError().

Ver también

IndicatorDelete(), IndicatorsTotal(), IndicatorName().

© 2000-2025, MetaQuotes Ltd.

IndicatorDelete

Borra el indicador dado de la ventana de gráfico especificada.

```
bool  IndicatorDelete(
   int            sub_win       // número de subventana
   const string   name          // nombre corto del indicador
   );
```

Parámetros

sub_win

[in] Número de subventana del gráfico. 0 significa la subventana del gráfico principal.

const name

[in] El nombre corto del indicador, cuyo nombre se establece en la propiedad INDICATOR_SHORTNAME por medio de la función IndicatorSetString(). Para obtener el nombre corto de un indicador se utiliza la función IndicatorName() .

Valor devuelto

Devuelve true si la operación de borrado del indicador se ejecuta correctamente. En caso contrario devuelve false. Para obtener los detalles del error utilizar la función GetLastError().

Nota

Si dos indicadores con dos nombres cortos idénticos existen en la subventana del gráfico, se eliminará el primero de la fila.

Si hay otros indicadores basados en los valores del indicador que ha sido borrado, tales indicadores también serán eliminados.

No confundir el nombre corto del indicador con el nombre de archivo que se especifica cuando se crea el indicador por medio de las funciones iCustom() y IndicatorCreate(). Si el nombre corto de un indicador no se establece explícitamente, entonces el nombre del archivo que contiene el código fuente del indicador se especificará durante la compilación.

Borrar un indicador de un gráfico no significa que sus cálculos sean eliminados de la memoria del terminal. Para liberar el manejador del indicador, utilizar la función IndicatorRelease().

El nombre corto del indicador debe estar bien formado. Se escribe en la propiedad INDICATOR_SHORTNAME por medio de la función IndicatorSetString(). Se recomienda que el nombre corto contenga valores de todos los parámetros de entrada del indicador, porque el indicador que se quiere eliminar del gráfico por medio de la función IndicatorDelete() se identifica con dicho nombre corto.

Ver también

IndicatorAdd(), IndicatorsTotal(), IndicatorName(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2025, MetaQuotes Ltd.

IndicatorsTotal

Devuelve el número de indicadores aplicados a la ventana de gráfico especificada.

```
int  IndicatorsTotal(
   int   sub_win     // número de subventana
   );
```

Parámetros

sub_win

[in] Número de subventana del gráfico. 0 significa la subventana del gráfico principal.

Valor devuelto

El número de indicadores en la ventana de gráfico especificada. Para obtener los detalles del error utilizar la función GetLastError().

Nota

La función permite la búsqueda a través de todos los indicadores adjuntados al gráfico. El número de ventanas del gráfico puede obtenerse a partir de la propiedad CHART_WINDOWS_TOTAL, por medio de la función GetInteger().

Ver también

IndicatorAdd(), IndicatorDelete(), IndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2025, MetaQuotes Ltd.

IndicatorName

Devuelve el nombre corto del indicador a partir del índice de la lista de indicadores, en la ventana de gráfico especificada.

```
string  IndicatorName(
   int    sub_win      // número de subventana
   int    index        // índice del indicador de la lista de indicadores añadida a la s
   );
```

Parámetros

sub_win

[in] Número de subventana del gráfico. 0 significa la subventana del gráfico principal.

index

[in] el índice del indicador de la lista de indicadores. La numeración de los indicadores empieza con el cero, así por ejemplo, el primer indicador de la lista tiene el índice 0. Para obtener el número de indicadores de la lista, utilizar la función IndicatorsTotal().

Valor devuelto

El nombre corto del indicador establecido en la propiedad INDICATOR_SHORTNAME por medio de la función IndicatorSetString(). Para obtener los detalles del error utilizar la función GetLastError().

Nota

No confundir el nombre corto del indicador con el nombre de archivo que se especifica cuando se crea el indicador por medio de las funciones iCustom() y IndicatorCreate(). Si el nombre corto del indicador no se establece explícitamente, entonces el nombre del archivo que contiene el código fuente del indicador será especificado durante la compilación.

El nombre corto del indicador debe estar bien formado. Se escribe en la propiedad INDICATOR_SHORTNAME por medio de la función IndicatorSetString(). Se recomienda que el nombre corto contenga valores de todos los parámetros de entrada del indicador, porque el indicador que se quiere eliminar del gráfico por medio de la función IndicatorDelete() se identifica con dicho nombre corto.

Ver también

IndicatorAdd(), IndicatorDelete, IndicatorsTotal, iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2025, MetaQuotes Ltd.

Navigate

Navega por el gráfico.

```
bool  Navigate(
   ENUM_CHART_POSITION  position,     // Posición
   int                  shift=0       // Desplazamiento
   )
```

Parámetros

position

[in] Valor de la enumeración ENUM_CHART_POSITION.

shift=0

[in] Número de barras a desplazar.

Valor devuelto

true si se ejecuta correctamente, false si no se ha navegado por el gráfico.

© 2000-2025, MetaQuotes Ltd.

Symbol

Obtiene el símbolo del gráfico

```
string  Symbol() const
```

Valor devuelto

Símbolo del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

Period

Obtiene el período del gráfico.

```
ENUM_TIMEFRAMES  Period() const
```

Valor devuelto

Período del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve 0.

© 2000-2025, MetaQuotes Ltd.

Redraw

Redibuja el gráfico asignado a la instancia de la clase.

```
void  Redraw()
```

© 2000-2025, MetaQuotes Ltd.

GetInteger

La función devuelve el valor de la correspondiente propiedad del objeto. La propiedad del objeto debe ser de tipo integer. Hay 2 variantes de la función

1. Devuelve inmediatamente el valor de la propiedad.

```
long  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,          // identificador de la propiedad
   int                          sub_window=0     // número de subventana
   ) const
```

2. Si se ejecuta correctamente, pone el valor de la propiedad en la variable especificada de tipo entero, pasada por referencia en el último parámetro.

```
bool  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,        // identificador de la propiedad
   int                          sub_window,     // número de subventana
   long&                        value           // aquí obtenemos el valor de la propi
   ) const
```

Parámetros

prop_id

[in] Identificador de la propiedad (ENUM_CHART_PROPERTY_INTEGER, enumeración).

sub_window

[in] Número de subventana del gráfico.

value

[in] Variable de tipo integer que recibe el valor de la propiedad solicitada.

Valor devuelto

Valor de la propiedad del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve -1.

En la segunda variante, la función devuelve true cuando la propiedad se mantiene y el valor se ha puesto en el valor de la variable; en caso contrario devuelve false. Para leer más acerca del error llamar a GetLastError().

© 2000-2025, MetaQuotes Ltd.

SetInteger

Establece un nuevo valor para la propiedad de tipo integer.

```
bool  SetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,      // identificador de la propiedad
   long                         value         // valor nuevo
   )
```

Parámetros

prop_id

[in] Identificador de la propiedad (ENUM_CHART_PROPERTY_INTEGER, enumeración).

value

[in] Nuevo valor de la propiedad.

Valor devuelto

true si se ejecuta correctamente, false si la propiedad de tipo integer no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

GetDouble

La función devuelve el valor de la correspondiente propiedad del objeto. La propiedad del objeto debe ser de tipo double. Hay 2 variantes de la función

1. Devuelve inmediatamente el valor de la propiedad.

```
double  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // identificador de la propiedad
   int                         sub_window=0     // número de subventana
   ) const
```

2. Si se ejecuta correctamente, pone el valor de la propiedad en la variable especificada de tipo double, pasado por referencia en el último parámetro.

```
bool  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,        // identificador de la propiedad
   int                         sub_window,     // número de subventana
   double&                     value           // aquí obtenemos el valor de la propie
   ) const
```

Parámetros

prop_id

[in] Identificador de la propiedad (ENUM_CHART_PROPERTY_DOUBLE enumeration).

sub_window

[in] Número de subventana del gráfico.

value

[in] Variable de tipo double que recibe el valor de la propiedad solicitada.

Valor devuelto

Valor de la propiedad del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve EMPTY_VALUE.

En la segunda variante, la función devuelve true cuando la propiedad se mantiene y el valor se ha puesto en el valor de la variable; en caso contrario devuelve false. Para leer más acerca del error llamar a GetLastError().

© 2000-2025, MetaQuotes Ltd.

SetDouble

Establece un nuevo valor para la propiedad de tipo double.

```
bool  SetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,      // identificador de la propiedad
   double                      value         // valor nuevo
   )
```

Parámetros

prop_id

[in] Identificador de la propiedad (enumeración ENUM_CHART_PROPERTY_DOUBLE).

value

[in] Nuevo valor de la propiedad.

Valor devuelto

true si se ejecuta correctamente, false si la propiedad de tipo double no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

GetString

La función devuelve el valor de la correspondiente propiedad del objeto. La propiedad del objeto debe ser de tipo string. Hay 2 variantes de la función

1. Devuelve inmediatamente el valor de la propiedad.

```
string  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id      // identificador de la propiedad
   ) const
```

2. Si se ejecuta correctamente, pone el valor de la propiedad en la variable especificada de tipo string, pasada por referencia en el último parámetro.

```
bool  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,     // identificador de la propiedad
   string&                     value        // aquí obtenemos el valor de la propiedad
   ) const
```

Parámetros

prop_id

[in] Identificador de la propiedad (ENUM_CHART_PROPERTY_STRING, enumeración).

sub_window

[in] Número de subventana del gráfico.

value

[in] Variable de tipo string que recibe el valor de la propiedad solicitada.

Valor devuelto

Valor de la propiedad del gráfico, asignado a la instancia de la clase. Si no hay ningún gráfico asignado, devuelve "".

En la segunda variante, la función devuelve true cuando la propiedad se mantiene y el valor se ha puesto en el valor de la variable; en caso contrario devuelve false. Para leer más acerca del error llamar a GetLastError().

© 2000-2025, MetaQuotes Ltd.

SetString

Establece un nuevo valor para la propiedad de tipo string.

```
bool  SetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,      // identificador de la propiedad
   string                      value         // nuevo valor de la propiedad
   )
```

Parámetros

prop_id

[in] Identificador de la propiedad (enumeración ENUM_CHART_PROPERTY_STRING).

value

[in] Nuevo valor de la propiedad.

Valor devuelto

true si se ejecuta correctamente, false si la propiedad de tipo string no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Cambia el símbolo y el período del gráfico asignado a la instancia de la clase.

```
bool  SetSymbolPeriod(
   const string      symbol_name,      // Símbolo
   ENUM_TIMEFRAMES   timeframe         // Período
   )
```

Parámetros

symbol_name

[in] Nuevo nombre del símbolo. NULL significa el símbolo del gráfico actual (donde se adjunta el experto).

timeframe

[in] Nuevo período de tiempo del gráfico (enumeración ENUM_TIMEFRAMES). 0 significa el período de tiempo actual.

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha sido cambiada.

© 2000-2025, MetaQuotes Ltd.

ApplyTemplate

Aplica en el gráfico la plantilla especificada.

```
bool  ApplyTemplate(
   const string  filename      // nombre de archivo de la plantilla
   )
```

Parámetros

filename

[in] Nombre de archivo de la plantilla.

Valor devuelto

true si se ejecuta correctamente, false si la plantilla no se puede aplicar.

© 2000-2025, MetaQuotes Ltd.

ScreenShot

Crea una captura de pantalla del gráfico especificado y la guarda en un archivo .gif

```
bool  ScreenShot(
   string           filename,                  // Nombre del archivo
   int              width,                     // Anchura
   int              height,                    // Altura
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // Tipo de alineación
   ) const
```

Parámetros

filename

[in] Nombre de archivo de la captura de pantalla.

width

[in] Anchura de la captura de pantalla en píxeles.

height

[in] Altura de la captura de pantalla en píxeles.

align_mode=ALIGN_RIGHT

[in] Modo de alineación, si la captura es estrecha.

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

WindowOnDropped

Obtiene el número de subventana del gráfico correspondiente al punto de caída del objeto (experto o script).

```
int  WindowOnDropped() const
```

Valor devuelto

Número de subventana del gráfico del punto de caída del objeto. 0 significa ventana del gráfico principal.

© 2000-2025, MetaQuotes Ltd.

PriceOnDropped

Obtiene la coordenada del precio correspondiente al punto de caída del objeto (experto o script).

```
double  PriceOnDropped() const
```

Valor devuelto

Coordenada del precio del punto de caída del objeto.

© 2000-2025, MetaQuotes Ltd.

TimeOnDropped

Obtiene las coordenadas temporales correspondientes al punto de caída del objeto (experto o script).

```
datetime  TimeOnDropped() const
```

Valor devuelto

Coordenada temporal del punto de caída del objeto.

© 2000-2025, MetaQuotes Ltd.

XOnDropped

Obtiene la coordenada X correspondiente al punto de caída del objeto (experto o script).

```
int  XOnDropped() const
```

Valor devuelto

Coordenada X del punto de caída del objeto.

© 2000-2025, MetaQuotes Ltd.

YOnDropped

Obtiene la coordenada Y correspondiente al punto de caída del objeto (experto o script).

```
int  YOnDropped() const
```

Valor devuelto

Coordenada Y del punto de caída del objeto.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda los parámetros del objeto en un archivo.

```
virtual bool  Save(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario abierto por la función FileOpen(...)

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Load

Carga los parámetros del objeto a partir de un archivo.

```
virtual bool  Load(
   int  file_handle      // Manejador del archivo
   )
```

Parámetros

file_handle

[in] manejador del archivo binario abierto por la función FileOpen(...)

Valor devuelto

true si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelveel identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo de objeto (0x1111 para CChart).

© 2000-2025, MetaQuotes Ltd.

Gráficos científicos

Biblioteca gráfica que contiene clases y funciones globales para dibujar rápidamente gráficos personalizados en el gráfico de precios. La biblioteca proporciona cómodas soluciones preparadas para construir ejes, curvas, y también métodos de acceso rápido al cambio de las propiedades generales de los gráficos personalizados.

Para comenzar a trabajar con la bilioteca, basta con leer el artículo ¡Visualice esto! La biblioteca gráfica en MQL5 como un análogo de plot en el lenguaje R.

La biblioteca gráfica se ubica en el catálogo de trabajo del terminal, en la carpeta Include\Graphics.

Clase	Descripción
CAxis	Clase para trabajar con ejes de coordenadas
ColorGenerator	Clase que establece el esquema de color por defecto
CCurve	Clase para trabajar con curvas
CGraphic	Clase base para crear gráficos personalizados

© 2000-2025, MetaQuotes Ltd.

GraphPlot

Función de dibujado rápido de curvas.

Versión para dibujar una curva según las coordenadas Y.

```
string  GraphPlot(
   const double      &y[],                  // coordenadas Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo de curva
   )
```

Nota

Para esta curva, los índices de la matriz Y actuarán como coordenadas X.

Versión para dibujar una curva según la pareja de coordenadas X e Y.

```
string  GraphPlot(
   const double      &x[],                  // coordenadas X
   const double      &y[],                  // coordenadas Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo de curva
   )
```

Versión para dibujar dos curvas según la pareja de coordenadas X e Y.

```
string  GraphPlot(
   const double      &x1[],                 // coordenadas X
   const double      &y1[],                 // coordenadas Y
   const double      &x2[],                 // coordenadas X
   const double      &y2[],                 // coordenadas Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo de curva
   )
```

Versión para dibujar tres curvas según la pareja de coordenadas X e Y.

```
string  GraphPlot(
   const double      &x1[],                 // coordenadas X
   const double      &y1[],                 // coordenadas Y
   const double      &x2[],                 // coordenadas X
   const double      &y2[],                 // coordenadas Y
   const double      &x3[],                 // coordenadas X
   const double      &y3[],                 // coordenadas Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo de curva
   )
```

© 2000-2025, MetaQuotes Ltd.

Versión para dibujar una curva según las coordenadas de los puntos CPoint2D.

```
string  GraphPlot(
   const CPoint2D   &points[],            // coordenadas de la curva
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

Versión para dibujar dos curvas según las coordenadas de los puntos CPoint2D.

```
string  GraphPlot(
   const CPoint2D   &points1[],           // coordenadas de la curva
   const CPoint2D   &points2[],           // coordenadas de la curva
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

Versión para dibujar tres curvas según las coordenadas de los puntos CPoint2D.

```
string  GraphPlot(
   const CPoint2D   &points1[],           // coordenadas de la curva
   const CPoint2D   &points2[],           // coordenadas de la curva
   const CPoint2D   &points3[],           // coordenadas de la curva
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

Versión para dibujar una curva según el puntero a la función CurveFunction.

```
string  GraphPlot(
   CurveFunction    function,             // puntero a la función
   const double     from,                 // valor inicial del argumento
   const double     to,                   // valor final del argumento
   const double     step,                 // incremento del argumento
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

Versión para dibujar dos curvas según los punteros a las funciones CurveFunction.

```
string  GraphPlot(
   CurveFunction    function1,            // puntero a la función
   CurveFunction    function2,            // puntero a la función
   const double     from,                 // valor inicial del argumento
   const double     to,                   // valor final del argumento
   const double     step,                 // incremento del argumento
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

© 2000-2025, MetaQuotes Ltd.

Versión para dibujar tres curvas según los punteros a las funciones CurveFunction.

```
string  GraphPlot(
   CurveFunction    function1,             // puntero a la función
   CurveFunction    function2,             // puntero a la función
   CurveFunction    function3,             // puntero a la función
   const double     from,                  // valor inicial del argumento
   const double     to,                    // valor final del argumento
   const double     step,                  // incremento del argumento
   ENUM_CURVE_TYPE  type=CURVE_POINTS      // tipo de curva
   )
```

Parámetros

&x[]

[in] Coordenadas X.

&y[]

[in] Coordenadas Y.

&x1[]

[in] Coordenadas X para la primera curva.

&y1[]

[in] Coordenadas X para la segunda curva.

&x2[]

[in] Coordenadas X para la segunda curva.

&y2[]

[in] Coordenadas Y para la segunda curva.

&x3[]

[in] Coordenadas X para la tercera curva.

&y3[]

[in] Coordenadas Y para la tercera curva.

&points[]

[in] Coordenadas de los puntos para la curva.

&points1[]

[in] Coordenadas de los puntos de la primera curva.

&points2[]

[in] Coordenadas de los puntos de la segunda curva.

&points3[]

[in] Coordenadas de los puntos de la tercera curva.

© 2000-2025, MetaQuotes Ltd.

function

[in] Puntero a la función CurveFunction.

function1

[in] Puntero a la primera función.

function2

[in] Puntero a la segunda función.

function3

[in] Puntero a la tercera función.

from

[in] Corresponde a la primera coordenada X.

to

[in] Corresponde a la última coordenada X.

step

[in] Parámetro para calcular las coordenadas X.

type=CURVE_POINTS

[in] Tipo de curva.

Valor devuelto

Nombre del recurso gráfico.

© 2000-2025, MetaQuotes Ltd.

CAxis

La clase CAxis es una clase auxiliar de la biblioteca gráfica para trabajar con ejes de coordenadas.

Descripción

La clase CAxis posibilita el almacenamiento y la obtención de diferentes parámetros de los ejes de coordenadas. En la clase se ha implementado la posibilidad de escalado automático y dinámico del eje de coordenadas.

Declaración

```
class CAxis
```

Encabezamiento

```
#include <Graphics\Axis.mqh>
```

Métodos de clase

Método	Descripción
AutoScale	Obtener/establecer la bandera de escalado automático
Min	Obtener/establecer el valor mínimo del eje
Max	Obtener/establecer el valor máximo del eje
Step	Retorna el valor del salto por el eje
Name	Obtener/establecer el nombre del eje
Color	Obtener/establecer el color del eje
ValuesSize	Obtener/establecer el tamaño de la fuente en el eje
ValuesWidth	Obtener/establecer la longitud máxima representada en el eje
ValuesFormat	Obtener/establecer el formato de las cifras en el eje
ValuesDateTimeMode	Retorna el formato de conversión de la fecha en una línea.
ValuesFunctionFormat	Retorna el puntero a la función que define el formato de exhibición de los valores en el eje.
ValuesFunctionFormatCBData	Retorna el puntero a un objeto en el que se puede contener información adicional para la conversión de los valores del eje.
NameSize	Obtener/establecer el tamaño de la fuente para el nombre del eje
ZeroLever	Obtener/establecer el valor "zero lever"

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
DefaultStep	Obtener/establecer el valor inicial del salto por el eje
MaxLabels	Obtener/establecer la cantidad máxima de cifras en el eje
MinGrace	Obtener/establecer el valor "umbral" para el mínimo del eje
MaxGrace	Obtener/establecer el valor "umbral" para el máximo del eje
SelectAxisScale	Realiza el escalado automático del eje.

© 2000-2025, MetaQuotes Ltd.

AutoScale (método Get)

Retorna la bandera que señala si hay que realizar el autoescalado.

```
bool  AutoScale()
```

Valor devuelto

Valor de la bandera.

Nota

true — ejecutar el autoescalado.

false — no ejecutar el autoescalado.

AutoScale (método Set)

Establece la bandera que señala si hay que realizar el autoescalado.

```
void  AutoScale(
   const bool  auto      // valor de la bandera
   )
```

Parámetros

auto

[in]

Nota

true — ejecutar el autoescalado.

false — no ejecutar el autoescalado.

© 2000-2025, MetaQuotes Ltd.

Min (método Get)

Devuelve el valor mínimo del eje.

```
double  Min()
```

Valor devuelto

Valor mínimo del eje.

Min (método Set)

Establece el valor mínimo del eje.

```
void  Min(
   const double  min      // valor mínimo
   )
```

Parámetros

min

[in] Valor mínimo.

© 2000-2025, MetaQuotes Ltd.

Max (método Get)

Devuelve el valor máximo del eje.

```
double  Max()
```

Valor devuelto

Valor máximo del eje.

Max (método Set)

Establece el valor máximo del eje del gráfico.

```
void  Max(
   const double  max       // valor máximo
   )
```

Parámetros

max

[in] Valor máximo del eje del gráfico.

© 2000-2025, MetaQuotes Ltd.

Step (método Get)

Retorna el valor del salto en el eje.

```
double  Step()
```

Valor devuelto

Valor del salto.

© 2000-2025, MetaQuotes Ltd.

Name (método Get)

Devuelve el nombre del eje.

```
string  Name()
```

Valor devuelto

Nombre del eje.

Name (método Set)

Establece el nombre del eje.

```
void  Name(
   const string  name      // nombre del eje
   )
```

Parámetros

name

[in] Nombre del eje.

© 2000-2025, MetaQuotes Ltd.

Color (método Get)

Devuelve el color del eje.

```
color  Color()
```

Valor devuelto

Color del eje.

Color (método Set)

Establece el color del eje.

```
void  Color(
   const color  clr      // color del eje
   )
```

Parámetros

clr

[in] Color del eje.

© 2000-2025, MetaQuotes Ltd.

ValuesSize (método Get)

Devuelve el tamaño de las cifras en el eje.

```
int  ValuesSize()
```

Valor devuelto

Tamaño de las cifras representadas en el eje.

ValuesSize (método Set)

Establece el tamaño de las cifras en el eje.

```
void  ValuesSize(
   const int  size      // tamaño de las cifras en el eje
   )
```

Parámetros

size

[in] Tamaño de las cifras en el eje

© 2000-2025, MetaQuotes Ltd.

ValuesWidth (método Get)

Devuelve la longitud máxima permitida en píxeles para representar cifras en el eje.

```
int  ValuesWidth()
```

Valor devuelto

Longitud de las cifras representadas en el eje en píxeles.

Nota

Si la longitud en píxeles para la cifra establecida supera la longitud máxima permitida de representación, se verá cortada y finalizada con puntos supensivos.

ValuesWidth (método Set)

Establece la longitud máxima permitida en píxeles para representar cifras en el eje.

```
void  ValuesWidth(
   const int  width      // distancia máxima permitida en píxeles
   )
```

Parámetros

width

[in] Distancia máxima permitida para representar cifras en el eje.

Nota

Si la longitud en píxeles para la cifra establecida supera la longitud máxima permitida de representación, se verá cortada y finalizada con puntos supensivos.

© 2000-2025, MetaQuotes Ltd.

ValuesFormat (método Get)

Devuelve el formato de representación de las cifras en el eje.

```
string  ValuesFormat()
```

Valor devuelto

Formato de representación de las cifras.

ValuesFormat (método Set)

Establece el formato de representación de las cifras en el eje.

```
void  ValuesFormat(
   const string  format      // formato de representación de las cifras en el eje
   )
```

Parámetros

format

[in] formato de representación de las cifras en el eje.

© 2000-2025, MetaQuotes Ltd.

ValuesDateTimeMode (Método Get)

Retorna el formato de conversión de la fecha en una línea.

```
int  ValuesDateTimeMode()
```

Valor devuelto

Formato de conversión de la fecha en una línea.

ValuesDateTimeMode (Метод Set)

Establece el formato de conversión de la fecha en una línea.

```
void  ValuesDateTimeMode(
   const int  mode      // formato de conversión de la fecha en una línea
   )
```

Parámetros

mode

[in] formato de conversión.

Nota

Dispone de más información sobre la conversión de formatos de fecha a línea en la descripción de la funciónTimeToString().

© 2000-2025, MetaQuotes Ltd.

ValuesFunctionFormat (Метод Get)

Retorna el puntero a la función que define el formato de exhibición de los valores en el eje.

```
DoubleToStringFunction  ValuesFunctionFormat()
```

Valor devuelto

Puntero a la función que define el formato de exhibición de los valores en el eje.

ValuesFunctionFormat (Método Set)

Establece el puntero a la función que define el formato de exhibición de los valores en el eje.

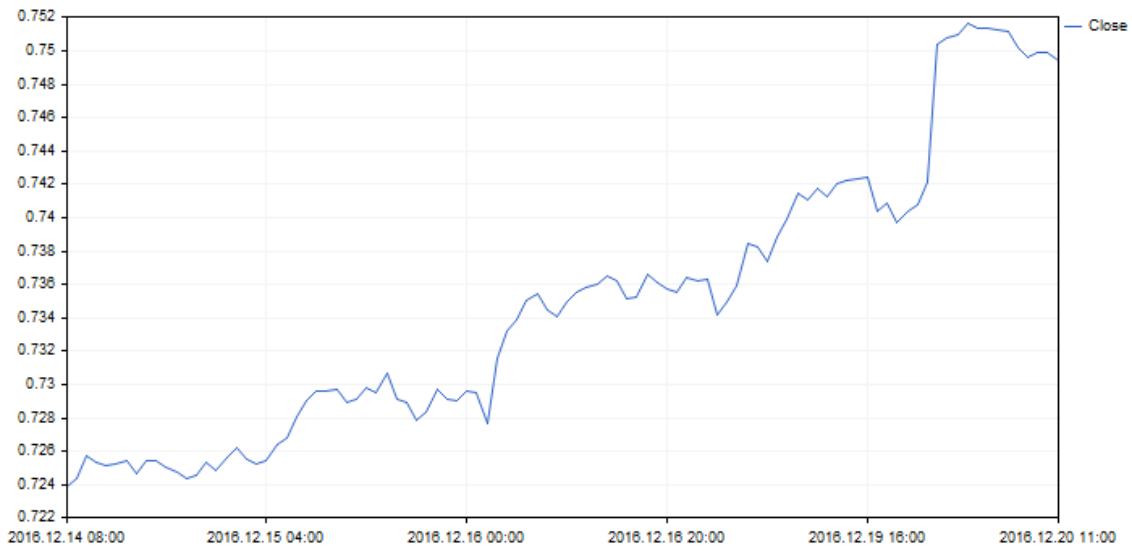
```
void  ValuesFunctionFormat(
   DoubleToStringFunction  func      // función para la conversión de valores numéric
   )
```

Parámetros

func

[in] Función de usuario para la conversión de valores numéricos a línea.

Ejemplo:



El formato de exhibición de los valores en el eje X se ha modificado con la ayuda del código siguiente:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                DateAxisGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//--- array for store values
double arrX[];
double arrY[];
//+------------------------------------------------------------------+
//| Custom function for create values on X-axis                      |
//+------------------------------------------------------------------+
string TimeFormat(double x,void *cbdata)
  {
   return(TimeToString((datetime)arrX[ArraySize(arrX)-(int)x-1]));
  }
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlRates rates[];
   CopyRates(Symbol(),Period(),0,100,rates);
   ArraySetAsSeries(rates,true);
   int size=ArraySize(rates);
   ArrayResize(arrX,size);
   ArrayResize(arrY,size);
   for(int i=0; i<size;++i)
     {
      arrX[i]=(double)rates[i].time;
      arrY[i]=rates[i].close;
     }
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"DateAxisGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"DateAxisGraphic");
     }
//--- create curve
   CCurve *curve=graphic.CurveAdd(arrY,CURVE_LINES);
//--- gets the X-axis
   CAxis *xAxis=graphic.XAxis();
//--- sets the X-axis properties
   xAxis.AutoScale(false);
   xAxis.Type(AXIS_TYPE_CUSTOM);
   xAxis.ValuesFunctionFormat(TimeFormat);
   xAxis.DefaultStep(20.0);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

ValuesFunctionFormatCBData (Метод Get)

Retorna el puntero a un objeto en el que se puede contener información adicional para la conversión de los valores del eje.

```
void*  ValuesFunctionFormatCBData()
```

Valor devuelto

Puntero a un objeto en el que se puede contener información adicional para la conversión de los valores del eje.

ValuesFunctionFormatCBData (Método Set)

Establece el puntero a un objeto de clase en el que se puede contener información adicional para la conversión de los valores del eje.

```
void  ValuesFunctionFormatCBData(
   void*  cbdata      // puntero al objeto de clase
   )
```

Parámetros

cbdata

[in] Puntero a un objeto de cualquier clase en el que se puede contener información adicional para la conversión de los valores del eje

© 2000-2025, MetaQuotes Ltd.

NameSize (método Get)

Devuelve el tamaño de la fuente para el nombre del eje.

```
int  NameSize()
```

Valor devuelto

Tamaño de la fuente para el nombre del eje.

NameSize (método Set)

Establece el tamaño de la fuente para el nombre del eje.

```
void  NameSize(
   const int  size      // tamaño de la fuente par el nombre del eje
   )
```

Parámetros

size

[in] Tamaño de la fuente para el nombre del eje.

© 2000-2025, MetaQuotes Ltd.

ZeroLever (método Get)

Devuelve el valor "zero lever".

```
double  ZeroLever()
```

Valor devuelto

"Zero lever".

Nota

Este valor se usa para definir cuándo el rango de la escala del eje debe ser ampliado para incluir el valor cero.

ZeroLever (método Set)

Establece el valor "zero lever".

```
void  ZeroLever(
   const double  value      // valor "zero lever"
   )
```

Parámetros

value

[in] Valor "zero lever".

Nota

Este valor se usa para definir cuándo el rango de la escala del eje debe ser ampliado para incluir el valor cero.

© 2000-2025, MetaQuotes Ltd.

DefaultStep (método Get)

Devuelve el valor inicial del salto en el eje.

```
double  DefaultStep()
```

Valor devuelto

Salto en el eje.

DefaultStep (método Set)

Establece el valor inicial del salto en el eje.

```
void  DefaultStep(
   const double  value      // salto en el eje
   )
```

Parámetros

value

[in] Valor inicial del salto en el eje.

© 2000-2025, MetaQuotes Ltd.

MaxLabels (método Get)

Devuelve el número máximo permitido de cifras representadas en el eje.

```
double  MaxLabels()
```

Valor devuelto

Número máximo de cifras en el eje.

MaxLabels (método Set)

Establece el número máximo permitido de cifras representadas en el eje.

```
void  MaxLabels(
   const double  value      // número máximo
   )
```

Parámetros

value

[in] Número máximo permitido de cifras representadas en el eje

© 2000-2025, MetaQuotes Ltd.

MinGrace (método Get)

Devuelve el valor de "tolerancia" aplicado al mínimo del eje.

```
double  MinGrace()
```

Valor devuelto

Valor de "tolerancia" para el mínimo del eje.

Nota

Este valor se expresa como una parte de la longitud total del eje. Por ejemplo, supongamos que los ejes se encuentaran en los límites de 4.0 a 16.0, entonces su longitud será igual a 12.0. Si MinGrace es igual a 0.1, entonces el 10% de la longitud del eje (o 1.2) se restará al valor del mínimo. Como resultado, el eje abarcará el intervalo de 2.8 a 16.0.

MinGrace (método Set)

Establece el valor de "tolerancia" aplicado al mínimo del eje.

```
void  MinGrace(
   const double  value      // valor de la "tolerancia"
   )
```

Parámetros

value

[in] Valor de "tolerancia" aplicado al mínimo del eje.

Nota

Este valor se expresa como una parte de la longitud total del eje. Por ejemplo, supongamos que los ejes se encuentaran en los límites de 4.0 a 16.0, entonces su longitud será igual a 12.0. Si MinGrace es igual a 0.1, entonces el 10% de la longitud del eje (o 1.2) se restará al valor del mínimo. Como resultado, el eje abarcará el intervalo de 2.8 a 16.0.

© 2000-2025, MetaQuotes Ltd.

MaxGrace (método Get)

Devuelve el valor de "tolerancia" aplicado al máximo del eje.

```
double  MaxGrace()
```

Valor devuelto

Valor de "tolerancia" para el máximo del eje.

Nota

Este valor se expresa como una parte de la longitud total del eje. Por ejemplo, supongamos que los ejes se encuentaran en los límites de 4.0 a 16.0, entonces su longitud será igual a 12.0. Si MaxGrace es igual a 0.1, entonces el 10% de la longitud del eje (o 1.2) se añadirá al valor del máximo. Como resultado, el eje abarcará el intervalo de 4.0 a 17.2.

MaxGrace (método Set)

Establece el valor de "tolerancia" aplicado al máximo del eje.

```
void  MaxGrace(
   const double  value      // valor de la "tolerancia"
   )
```

Parámetros

value

[in] Valor de tolerancia aplicado al máximo del eje.

Nota

Este valor se expresa como una parte de la longitud total del eje. Por ejemplo, supongamos que los ejes se encuentaran en los límites de 4.0 a 16.0, entonces su longitud será igual a 12.0. Si MinGrace es igual a 0.1, entonces el 10% de la longitud del eje (o 1.2) se restará al valor del mínimo. Como resultado, el eje abarcará el intervalo de 2.8 a 16.0.

© 2000-2025, MetaQuotes Ltd.

SelectAxisScale

Realiza el escalado automático del eje.

```
void  SelectAxisScale()
```

© 2000-2025, MetaQuotes Ltd.

CColorGenerator

La clase CColorGenerator es una clase auxiliar de la biblioteca gráfica para trabajar con la paleta de colores.

Descripción

La clase CColorGenerator contiene una paleta inicial de colores utilizados por defecto para las curvas (si el usuario no ha indicado un color).

Si todos los colores de la paleta han sido usados, comenzará la generación automática de nuevos colores, y la paleta se rellenará otra vez.

Declaración

```
class CColorGenerator
```

Encabezamiento

```
#include <Graphics\ColorGenerator.mqh>
```

Métodos de la clase

Método	Descripción
Next	Devuelve el siguiente color de la paleta
Reset	Lleva al generador a su estado inicial

© 2000-2025, MetaQuotes Ltd.

Next

Devuelve el siguiente color de la paleta.

```
uint  Next()
```

Valor devuelto

Color.

Nota

Si todos los colores de la paleta han sido seleccionados, comenzará la generación automática de nuevos colores, que sustituirán a los antiguos en la paleta.

© 2000-2025, MetaQuotes Ltd.

Reset

Lleva al generador a su estado inicial.

```
void  Reset()
```

© 2000-2025, MetaQuotes Ltd.

CCurve

La clase CCurve es una clase para trabajar con las propiedades de las curvas creadas en el gráfico.

Descripción

La clase CCurve posibilita la instalación, el almacenamiento y la obtención de coordenadas y diferentes propiedades de las curvas al trabajar con la clase CGraphic.

Se han previsto tres modos de dibujado de curvas: con puntos, con líneas y con histograma. Para cada modo de dibujado, en la clase se han implementado parámetros aparte.

Declaración

```
class CCurve : public CObject
```

Encabezamiento

```
#include <Graphics\Curve.mqh>
```

Jerarquía de herencia

CObject

CCurve

Métodos de clase

Método	Descripción
Type	Devuelve el tipo de curva
Name	Devuelve el nombre de la curva
Color	Devuelve el color de la curva
XMax	Devuelve el valor máximo de la función en el eje X
XMin	Devuelve el valor mínimo de la función en el eje X
YMax	Devuelve el valor máximo de la función en el eje Y
YMin	Devuelve el valor mínimo de la función en el eje Y
Size	Devuelve el número de puntos que definen la curva
PointsSize	Obtener/establecer el tamaño lineal de los puntos que definen la curva
PointsFill	Obtener/establecer la bandera de rellenado de los puntos que definen la curva
PointsColor	Obtener/establecer el color de rellenado de los puntos
GetX	Recibe en una matriz los valores de X de todos los puntos de la curva

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
GetY	Recibe en una matriz los valores de Y de todos los puntos de la curva
LinesStyle	Obtener/establecer el estilo de la línea al dibujar la curva con líneas
LinesIsSmooth	Obtener/establecer la bandera de suavizado al dibujar con líneas
LinesSmoothTension	Obtener/establecer el parámetro de suavizado de la curva al dibujar con líneas
LinesSmoothStep	Obtener/establecer la longitud de las líneas que se aproximan para el suavizado al dibujar con líneas
LinesWidth	Obtener/establecer la anchura de las líneas al dibujar la curva con líneas.
HistogramWidth	Obtener/establecer la anchura de las columnas al dibujar con histograma
CustomPlotCBData	Obtener/establecer el puntero al objeto que puede utilizarse en el modo personalizado de dibujado de la curva.
CustomPlotFunction	Obtener/establecer el puntero a la función que implementa el modo personalizado de dibujado de la curva.
PointsType	Obtener/establecer la bandera que indica el tipo de puntos utilizados al dibujar la curva con puntos.
StepsDimension	Obtener/establecer el valor que indica la dimensión en la que se hace el salto al dibujar la curva de forma escalonada.
TrendLineCoefficients	Obtener/establecer los coeficientes de la línea de tendencia para grabarlos en la matriz.
TrendLineColor	Obtener/establecer el color de la línea de tendencia para la curva.
TrendLineVisible	Obtener/establecer la bandera de visibilidad de la línea de tendencia.
Update	Actualiza las coordinadas de la curva.
Visible	Obtener/establecer la bandera que indica si será visible la función en el gráfico.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Compare

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el tipo de curva.

```
ENUM_CURVE_TYPE  Type()
```

Valor devuelto

Tipo de curva.

© 2000-2025, MetaQuotes Ltd.

Name

Devuelve el nombre de la curva.

```
string  Name()
```

Valor devuelto

Nombre de la curva.

© 2000-2025, MetaQuotes Ltd.

Color

Devuelve el color de la curva.

```
uint  Color()
```

Valor devuelto

Color de la curva.

© 2000-2025, MetaQuotes Ltd.

XMax

Devuelve el valor máximo para la función en el eje X (solo números reales).

```
double  XMax()
```

Valor devuelto

Número real máximo entre todos los argumentos para la función dada.

© 2000-2025, MetaQuotes Ltd.

XMin

Devuelve el valor mínimo para la función en el eje X (solo valores reales).

```
double  XMin()
```

Valor devuelto

Número real mínimo entre todos los argumentos para la función dada.

© 2000-2025, MetaQuotes Ltd.

YMax

Devuelve el valor máximo para la función en el eje Y (solo números reales).

```
double  YMax()
```

Valor devuelto

Valor máximo para la función dada en el eje Y (solo números reales).

© 2000-2025, MetaQuotes Ltd.

YMin

Devuelve el valor mínimo para la función en el eje Y (solo números reales).

```
double  YMin()
```

Valor devuelto

Valor mínimo para la función dada en el eje Y (solo números reales).

© 2000-2025, MetaQuotes Ltd.

Size

Devuelve el número de puntos que definen esta curva.

```
int  Size()
```

Valor devuelto

Número de puntos por el que se define la curva.

© 2000-2025, MetaQuotes Ltd.

PointsSize (método Get)

Devuelve el tamaño lineal de los puntos que definen esta curva al dibujarla con puntos, en píxeles.

```
int  PointsSize()
```

Valor devuelto

Tamaño de los puntos que definen le curva, en píxeles.

PointsSize (método Set)

Establece el tamaño de los puntos que definen esta curva al dibujarla con puntos, en píxeles.

```
void  PointsSize(
   const int  size      // tamaño de los puntos en píxeles
   )
```

Parámetros

size

[in] Tamaño de los puntos que definen esta curva, en píxeles.

© 2000-2025, MetaQuotes Ltd.

PointsFill (método Get)

Devuelve la bandera que indica si hay que ejecutar el rellenado de los puntos que definen la curva al dibujar con puntos.

```
bool  PointsFill ()
```

Valor devuelto

Valor de la bandera.

Nota

true — ejecutar rellenado

false — no ejecutar rellenado

PointsFill (método Set)

Establece la bandera que indica si hay que ejecutar el rellenado de los puntos que definen la curva al dibujar con puntos.

```
void  PointsFill(
   const bool  fill      // valor de la bandera
   )
```

Parámetros

fill

[in] Valor de la bandera.

Nota

true — ejecutar rellenado

false — no ejecutar rellenado

© 2000-2025, MetaQuotes Ltd.

PointsColor (método Get)

Devuelve el color de rellenado de los puntos.

```
uint  PointsColor ()
```

Valor devuelto

Color de rellenado de los puntos que definen esta curva.

PointsColor (método Set)

Establece el color de rellenado de los puntos.

```
void  PointsColor(
   const uint  clr      //color de rellenado de los puntos
   )
```

Parámetros

clr

[in] Color de rellenado de los puntos que definen esta curva.

© 2000-2025, MetaQuotes Ltd.

GetX

Recibe en una matriz los valores X de todos los puntos de la curva.

```
void  GetX(
   double&  x[]      // matriz para escribir los valores X
   )
```

Parámetros

x[]

[out] Matriz para obtener los valores X de todos los puntos de la curva.

Nota

Cada punto de la curva se establece con la pareja de valores X e Y. Estos valores no son coordenadas en píxeles para dibujar en la clase CGraphic.

© 2000-2025, MetaQuotes Ltd.

GetY

Recibe en una matriz los valores Y de todos los puntos de la curva.

```
void  GetY(
   double&  y[]      // matriz para escribir los valores Y
   )
```

Parámetros

y[]

[out] Matriz para obtener los valores Y de todos los puntos de la curva.

Nota

Cada punto de la curva se establece con la pareja de valores X e Y. Estos valores no son coordenadas en píxeles para dibujar en la clase CGraphic.

© 2000-2025, MetaQuotes Ltd.

LinesStyle (método Get)

Devuelve el estilo de la línea al dibujar con líneas la curva

```
ENUM_LINE_STYLE  LinesStyle()
```

Valor devuelto

Estilo de la línea.

LinesStyle (método Set)

Establece el estilo de las líneas al dibujar con líneas la curva

```
void  LinesStyle (
   ENUM_LINE_STYLE  style      // estilo de la línea
   )
```

Parámetros

style

[in] Estilo de la línea.

© 2000-2025, MetaQuotes Ltd.

LinesIsSmooth (método Get)

Devuelve la bandera que determina si hay que realizar el suavizado tras dibujar la curva con líneas.

```
bool  LinesIsSmooth()
```

Valor devuelto

Valor de la bandera

Nota

true — ejecutar suavizado

false — no ejecutar suavizado

LinesIsSmooth (método Set)

Establece la bandera que determina si hay que realizar el suavizado tras dibujar la curva con líneas.

```
void  LinesIsSmooth(
   const bool  smooth      // valor de la bandera
   )
```

Parámetros

smooth

[in] Valor de la bandera

Nota

true — ejecutar suavizado

false — no ejecutar suavizado

© 2000-2025, MetaQuotes Ltd.

LinesSmoothTension (método Get)

Devuelve el parámetro de suavizado de la curva al dibujar con líneas

```
double  LinesSmoothTension()
```

Valor devuelto

Valor del parámetro de suavizado

Nota

El valor tension se encuentra en el rango (0.0; 1.0].

LinesSmoothTension (método Set)

Establece el parámetro de suavizado de la curva al dibujar con líneas

```
void  LinesSmoothTension(
   const double  tension      // valor del parámetro
   )
```

Parámetros

tension

[in] Valor del parámetro de suavizado.

Nota

El valor tension se encuentra en el rango (0.0; 1.0].

© 2000-2025, MetaQuotes Ltd.

LinesSmoothStep (método Get)

Devuelve la longitud de las líneas aproximativas para el suavizado al dibujar con líneas.

```
double  LinesSmoothStep()
```

Valor devuelto

Longitud de las líneas aproximativas en píxeles.

LinesSmoothStep (método Set)

Establece la longitud de las líneas aproximativas para el suavizado al dibujar con líneas.

```
void  LinesSmoothStep(
   const double  step      // longitud de las líneas
   )
```

Parámetros

step

[in] Longitud de las líneas aproximativas

© 2000-2025, MetaQuotes Ltd.

LinesEndStyle (Método Set)

Retorna la bandera que indica el estilo de dibujado de los extremos de las líneas al dibujar la curva con líneas.

```
ENUM_LINE_END  LinesEndStyle()
```

Valor devuelto

Valor de la bandera que indica el estilo de dibujado de los extremos de las líneas al dibujar la curva con líneas.

LinesEndStyle (Método Get)

Establece la bandera que indica el estilo de dibujado de los extremos de las líneas al dibujar la curva con líneas.

```
void  LinesEndStyle(
   ENUM_LINE_END  end_style      // valor de la bandera
   )
```

Parámetros

end_style

[in] Valor de la bandera que indica el estilo de los extremos de la línea al dibujar la curva con líneas.

© 2000-2025, MetaQuotes Ltd.

LinesWidth (Método Get)

Retorna la anchura de las líneas al dibujar la curva con líneas.

```
int  LinesWidth()
```

Valor devuelto

Anchura de las líneas.

LinesWidth (Método Set)

Establece la anchura de las líneas al dibujar la curva con líneas.

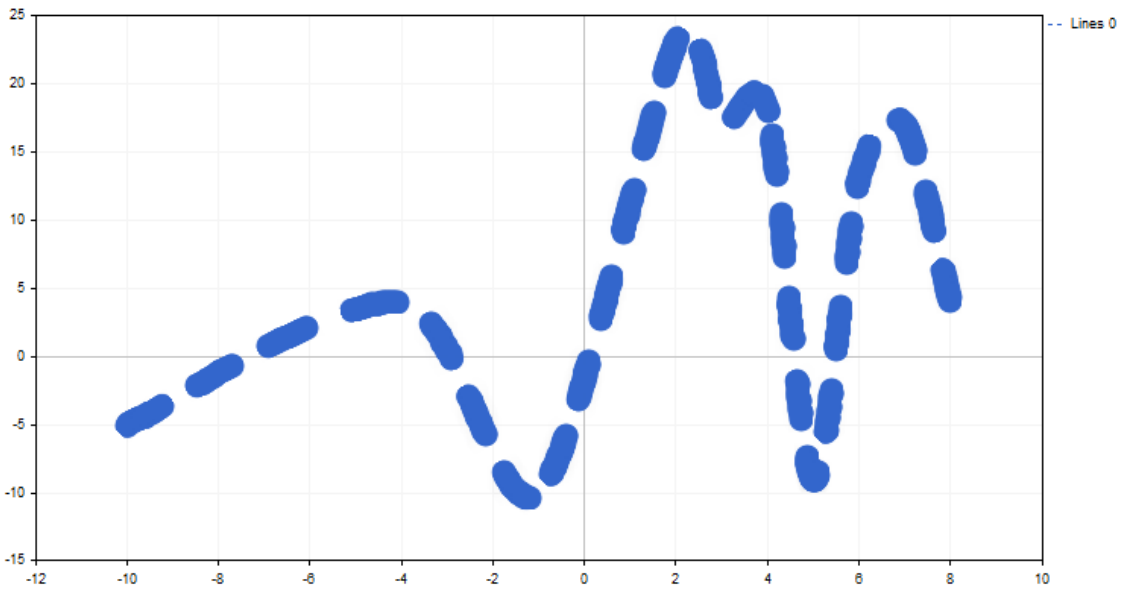
```
void  LinesWidth(
   const int  width      // anchura de las líneas
   )
```

Parámetros

width

[in] Anchura de las líneas al dibujar la curva con líneas.

Ejemplo:



La anchura de las líneas se ha modificado con la ayuda del siguiente código:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]= { -100,-40,-10,20,30,40,50,60,70,80,120 };
   double y[]= { -5,4,-10,23,17,18,-9,13,17,4,9 };
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"ThickLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"ThickLineGraphic");
     }
//--- create curve
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_LINES);
//--- sets the curve properties
   curve.LinesSmooth(true);
   curve.LinesStyle(STYLE_DASH);
   curve.LinesEndStyle(LINE_END_ROUND);
   curve.LinesWidth(10);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

HistogramWidth (método Get)

Devuelve la anchura de las columnas al dibujar con histograma.

```
int  HistogramWidth()
```

Valor devuelto

Anchura de las columnas en píxeles.

HistogramWidth (método Set)

Establece la anchura de las columnas al dibujar con histograma.

```
void  HistogramWidth(
   const int  width      // anchura de las columnas
   )
```

Parámetros

width

[in] Anchura de las columnas en píxeles.

© 2000-2025, MetaQuotes Ltd.

CustomPlotCBData (Método Get)

Retorna el puntero a un objeto que puede utilizarse en el modo personalizado de dibujado de la curva.

```
void*  CustomPlotCBData()
```

Valor devuelto

Puntero a un objeto para el modo personalizado de dibujado de la curva.

CustomPlotCBData (Método Set)

Establece el puntero al objeto de clase que se usará en el modo personalizado de dibujado de la curva.

```
void  CustomPlotCBData(
   void*  cbdata      // puntero al objeto
   )
```

Parámetros

cbdata

[in] Puntero a un objeto de cualquier clase que se usará en el modo personalizado de dibujado de la curva

© 2000-2025, MetaQuotes Ltd.

CustomPlotFunction (Método Get)

Retorna el puntero a la función que implementa el modo personalizado de dibujado de la curva.

```
PlotFucntion  CustomPlotFunction()
```

Valor devuelto

Puntero a la función que implementa el modo personalizado de dibujado de la curva.

CustomPlotFunction (Método Set)

Establece el puntero a la función que implementa el modo personalizado de dibujado de la curva.

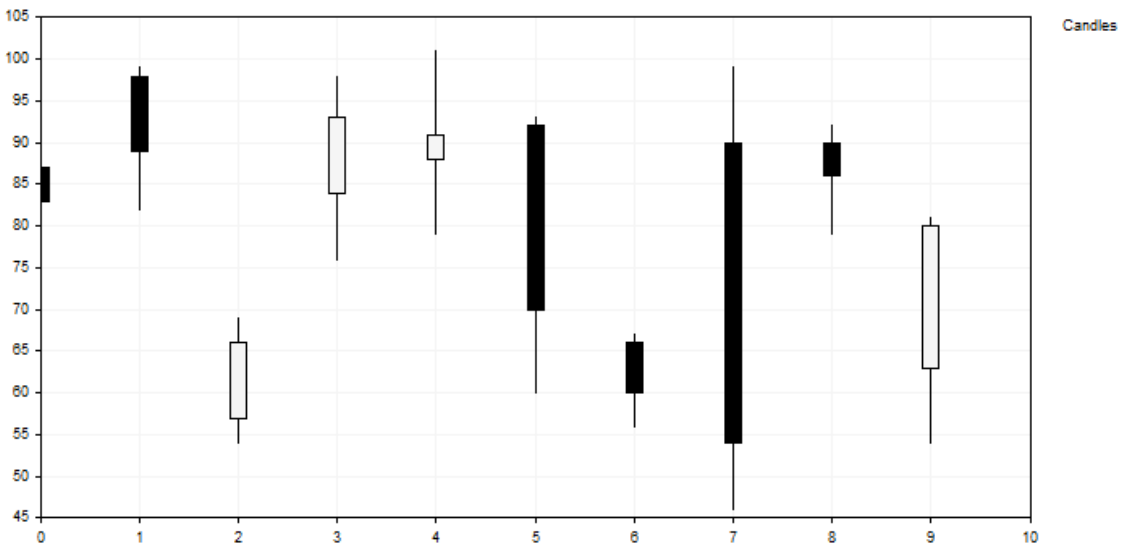
```
void  CustomPlotFunction(
   PlotFucntion  func      // puntero a la función
   )
```

Parámetros

func

[in] Puntero a la función que implementa el modo personalizado de dibujado de la curva.

Ejemplo:



Esta curva de barras ha sido construida con la ayuda del siguiente código:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Class CCandle                                                    |
//| Usage: class to represent the candle                             |
//+------------------------------------------------------------------+
class CCandle: public CObject
  {
private:
   double            m_open;
   double            m_close;
   double            m_high;
   double            m_low;
   uint              m_clr_inc;
   uint              m_clr_dec;
   int               m_width;

public:
                     CCandle(const double open,const double close,const double high,co
                                                    const int width,const uint clr_
                    ~CCandle(void);
   double            OpenValue(void)            const { return(m_open);     }
   double            CloseValue(void)           const { return(m_close);    }
   double            HigthValue(void)           const { return(m_high);     }
   double            LowValue(void)             const { return(m_low);      }
   uint              CandleColorIncrement(void) const { return(m_clr_inc);  }
   uint              CandleColorDecrement(void) const { return(m_clr_dec);  }
   int               CandleWidth(void)          const { return(m_width);    }
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CCandle::CCandle(const double open,const double close,const double high,const double l
                                 const int width,const uint clr_inc=0x000000,const uin
                                 m_open(open),m_close(close),m_high(high),m_low(low),
                                 m_clr_inc(clr_inc),m_clr_dec(clr_dec),m_width(width)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CCandle::~CCandle(void)
  {
  }
//+------------------------------------------------------------------+
//| Custom method for plot candles                                   |
//+------------------------------------------------------------------+
void PlotCandles(double &x[],double &y[],int size,CGraphic *graphic,CCanvas *canvas,vo
  {
//--- check obj
   CArrayObj *candles=dynamic_cast<CArrayObj*>(cbdata);
   if(candles==NULL || candles.Total()!=size)
      return;
//--- plot candles
   for(int i=0; i<size; i++)
     {
      CCandle *candle=dynamic_cast<CCandle*>(candles.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      if(candle==NULL)
         return;
      //--- primary calculate
      int xc=graphic.ScaleX(x[i]);
      int width_2=candle.CandleWidth()/2;
      int open=graphic.ScaleY(candle.OpenValue());
      int close=graphic.ScaleY(candle.CloseValue());
      int high=graphic.ScaleY(candle.HigthValue());
      int low=graphic.ScaleY(candle.LowValue());
      uint clr=(open<=close) ? candle.CandleColorIncrement() :  candle.CandleColorDecr
      //--- plot candle
      canvas.LineVertical(xc,high,low,0x000000);
      //--- plot candle real body
      canvas.FillRectangle(xc+width_2,open,xc-width_2,close,clr);
      canvas.Rectangle(xc+width_2,open,xc-width_2,close,0x000000);
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int count=10;
   int width=10;
   double x[];
   double y[];
   ArrayResize(x,count);
   ArrayResize(y,count);
   CArrayObj candles();
   double max=0;
   double min=0;
//--- create values
   for(int i=0; i<count; i++)
     {
      x[i] = i;
      y[i] = i;
      //--- calculate values
      double open=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double close=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double high=MathRound(MathMax(open,close)+(MathRand()/32767.0)*10.0);
      double low=MathRound(MathMin(open,close) -(MathRand()/32767.0)*10.0);
      //--- find max and min
      if(i==0 || max<high)
         max=high;
      if(i==0 || min>low)
         min=low;
      //--- create candle
      CCandle *candle=new CCandle(open,close,high,low,width);
      candles.Add(candle);
     }
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"CandleGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"CandleGraphic");
     }
//--- create curve
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_CUSTOM,"Candles");
//--- sets the curve properties
   curve.CustomPlotFunction(PlotCandles);
   curve.CustomPlotCBData(GetPointer(candles));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- sets the graphic properties
   graphic.YAxis().Max((int)max);
   graphic.YAxis().Min((int)min);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

PointsType (Método Get)

Retorna la bandera que indica el tipo de puntos utilizados al dibujar la curva con puntos.

```
ENUM_POINT_TYPE  PointsType()
```

Valor devuelto

Valor de la bandera que indica el tipo de puntos.

PointsType (Método Set)

Establece la bandera que indica el tipo de puntos utilizados al dibujar la curva con puntos.

```
void  PointsType(
   ENUM_POINT_TYPE  type      // valor de la bandera
   )
```

Parámetros

type

[in] Valor de la bandera que indica el tipo de puntos utilizados al dibujar la curva con puntos.

© 2000-2025, MetaQuotes Ltd.

StepsDimension (Método Get)

Retorna el valor que indica la dimensión en la que se hace el salto al dibujar la curva de forma escalonada.

```
int  StepsDimension()
```

Valor devuelto

Dimensión en la que se hace el salto al dibujar la curva de forma escalonada.

StepsDimension (Método Set)

Establecer el valor que indica la dimensión en la que se hace el salto al dibujar la curva de forma escalonada.

```
void  StepsDimension(
   const int  dimension      // dimensión
   )
```

Parámetros

dimension

[in] Dimensión (puede ser o bien 0, o bien 1).

Nota

0 — x (primero se dibuja la línea horizontal, después la vertical).

1 — y (primero se dibuja la línea vertical, después la horizontal).

© 2000-2025, MetaQuotes Ltd.

TrendLineCoefficients (Método Get)

Retorna los coeficientes de la línea de tendencia para grabarlos en la matriz.

```
double&  TrendLineCoefficients()
```

Valor devuelto

Coeficientes de la línea de tendencia.

TrendLineCoefficients (Método Set)

Obtiene los coeficientes de la línea de tendencia para grabarlos en la matriz.

```
void  TrendLineCoefficients(
   double&  coefficients[]      // matriz para la grabación de coeficientes
   )
```

Parámetros

coefficients[]

[out] Matriz para la grabación de los coeficientes.

© 2000-2025, MetaQuotes Ltd.

TrendLineColor (Método Get)

Retorna el color de la línea de tendencia para la curva.

```
uint  TrendLineColor()
```

Valor devuelto

Color de la línea de tendencia.

TrendLineColor (Método Set)

Establece el color de la línea de tendencia para la curva.

```
void  TrendLineColor(
   const uint  clr      // color de la línea de tendencia
   )
```

Parámetros

clr

[in] Color de la línea

© 2000-2025, MetaQuotes Ltd.

TrendLineVisible (Método Get)

Retorna la bandera que define la visibilidad de la línea de tendencia.

```
bool  TrendLineVisible()
```

Valor devuelto

Valor de la bandera que indica si la línea de tendencia es visible.

TrendLineVisible (Método Set)

Establece la bandera que define la visibilidad de la línea de tendencia.

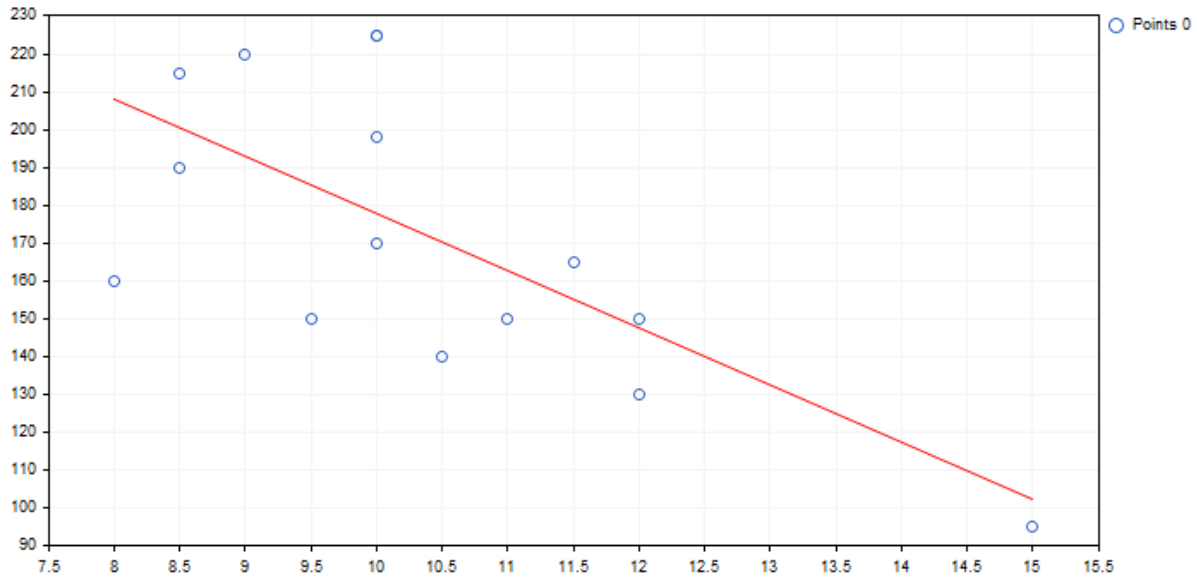
```
void  TrendLineVisible(
   const bool  visible      // valor de la bandera
   )
```

Parámetros

visible

[in] Valor de la bandera que define la visibilidad de la línea de tendencia.

Ejemplo:



Código para construir la línea mostrada más arriba y representar la misma en el gráfico:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              TrendLineGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]={12.0,11.5,11.0,12.0,10.5,10.0,9.0,8.5,10.0,8.5,10.0,8.0,9.5,10.0,15.0};
   double y[]={130.0,165.0,150.0,150.0,140.0,198.0,220.0,215.0,225.0,190.0,170.0,160.0
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"TrendLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"TrendLineGraphic");
     }
//--- create curve
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_POINTS);
//--- sets the curve properties
   curve.TrendLineVisible(true);
   curve.TrendLineColor(ColorToARGB(clrRed));
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Actualiza las coordinadas de la curva.

Versión para trabajar con la coordenada Y. Las coordenadas X en este caso serán los índices de la matriz transmitida.

```
void  Update(
   const double&  y[]   // Coordenadas Y
   )
```

Versión para trabajar con la pareja de coordenadas X e Y

```
void  Update(
   const double&  x[],     // Coordenadas X
   const double&  y[]      // Coordenadas Y
   )
```

Versión para trabajar con los puntos CPoint2D.

```
void  Update(
   const CPoint2D&  points[]      // Coordenadas de la curva
   )
```

Versión para trabajar con el puntero a la función CurveFunction.

```
void  Update(
   CurveFunction  function,     // puntero a la función que describe la curva
   const double   from,         // valor inicial del argumento de la función
   const double   to,           // valor final del argumento de la función
   const double   step          // incremento del argumento
   )
```

Parámetros

x[]

[in] Coordenadas X.

y[]

[in] Coordenadas Y.

points[]

[in] Coordenadas de la curva.

function

[in] Puntero a la función que describe la curva

from

[in] Valor inicial del argumento de la función

to

[in] Valor final del argumento de la función

step

© 2000-2025, MetaQuotes Ltd.

[in] Incremento del argumento

© 2000-2025, MetaQuotes Ltd.

Visible (Método Get)

Retorna la bandera que indica si será visible la función en el gráfico.

```
void  Visible(
   const bool  visible      //
   )
```

Valor devuelto

Valor de la bandera de visibilidad de la función en el gráfico.

Visible (Método Set)

Establece la bandera que indica si será visible la función en el gráfico.

```
void  Visible(
   const bool  visible      // valor de la bandera
   )
```

Parámetros

visible

[in] Valor de la bandera de visibilidad de la función en el gráfico.

© 2000-2025, MetaQuotes Ltd.

CGraphic

La clase CGraphic es la clase base para crear gráficos personalizados.

Descripción

La clase CGraphic proporciona múltiples posibilidades para trabajar con gráficos personalizados.

En la clase se guardan los principales elementos del gráfico, se indican sus parámetros y se implementa su dibujado.

Asimismo, esta clase guarda las curvas para un gráfico dado y proporciona diferentes variantes de representación de las mismas.

Declaración

```
class CGraphic
```

Encabezamiento

```
#include <Graphics\Graphic.mqh>
```

Métodos de clase

Método	Descripción
Create	Crea un recurso gráfico vinculado a un objeto del gráfico
Destroy	Elimina gráficos y borra recursos gráficos
Update	Representa en la pantalla los cambios realizados
ChartObjectName	Retorna el nombre del objeto vinculado al gráfico
ResourceName	Retorna el nombre del recurso gráfico
XAxis	Retorna el puntero al eje X
YAxis	Retorna el puntero al eje Y
GapSize	Obtener/establecer el tamaño de los márgenes entre los elementos del gráfico
BackgroundColor	Obtener/establecer el color del fondo
BackgroundMain	Obtener/establecer el texto del encabezamiento del gráfico
BackgroundMainSize	Obtener/establecer el tamaño de la fuente para el encabezamiento
BackgroundMainColor	Obtener/establecer el color del encabezamiento para gráfico
BackgroundSub	Obtener/establecer el texto del subtítulo del gráfico

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
BackgroundSubSize	Obtener/establecer el tamaño de la fuente para el encabezamiento
BackgroundSubColor	Obtener/establecer el color del subtítulo para el gráfico
GridLineColor	Obtener/establecer el color de las líneas de la cuadrícula
GridBackgroundColor	Obtener/establecer el color del fondo de la cuadrícula
GridCircleRadius	Obtener/establecer el radio de los puntos en los nodos de la cuadrícula
GridCircleColor	Obtener/establecer el color de los puntos en los nodos de la cuadrícula
GridHasCircle	Obtener/establecer la bandera de dibujado de los puntos en los nodos de la cuadrícula
GridAxisLineColor	Retorna el valor del color de los ejes reales del gráfico.
HistoryNameWidth	Obtener/establecer la longitud máxima permitida para representar el nombre de la curva
HistoryNameSize	Obtener/establecer el tamaño de la fuente para el nombre de la curva
HistorySymbolSize	Obtener/establecer el tamaño de los símbolos de los signos convencionales
TextAdd	Añade texto al gráfico
LineAdd	Añade una línea al gráfico
CurveAdd	Crea una curva y la añade al gráfico
CurvePlot	Dibuja una curva creada con anterioridad conforme a un índice
CurvePlotAll	Dibuja todas las curvas creadas con anterioridad
CurveGetByIndex	Obtiene una curva según el índice establecido
CurveGetByName	Obtiene una curva según el nombre establecido
CurveRemoveByIndex	Elimina una curva según el índice establecido.
CurveRemoveByName	Elimina una curva según el nombre establecido.
CurvesTotal	Retorna el número de curvas para el gráfico dado.
MarksToAxisAdd	Añade una marca de la escala al eje del gráfico
MajorMarkSize	Obtener/establecer el tamaño de las "muescas" de la escala en el eje del gráfico

© 2000-2025, MetaQuotes Ltd.

Método	Descripción
FontSet	Establecer los parámetros de la fuente actual
FontGet	Obtener los parámetros de la fuente actual
Attach	Obtener/crear un recurso gráfico y vincularlo al ejemplar de la clase CGraphic
CalculateMaxMinValues	Realiza el cálculo (recálculo) de los valores máximos y mínimos del gráfico en los dos ejes.
Height	Retorna la altura del gráfico en píxeles.
IndentDown	Obtener/establecer el margen del gráfico con respecto al límite inferior.
IndentLeft	Obtener/establecer el margen del gráfico con respecto al límite izquierdo.
IndentRight	Obtener/establecer el margen del gráfico con respecto al límite derecho.
IndentUp	Obtener/establecer el margen del gráfico con respecto al límite superior.
Redraw	Redibuja el gráfico.
ResetParameters	Resetea los parámetros necesarios para redibujar el gráfico.
ScaleX	Escala el valor en el eje X.
ScaleY	Escala el valor en el eje Y.
SetDefaultParameters	Da a los parámetros del gráfico los valores por defecto.
Width	Retorna la anchura del gráfico en píxeles.

© 2000-2025, MetaQuotes Ltd.

Create

Crea un recurso gráfico ligado a un objeto del gráfico.

```
bool  Create(
   const long    chart,      // identificador del gráfico de precio
   const string  name,       // nombre
   const int     subwin,     // número de subventana
   const int     x1,         // coordenada x1
   const int     y1,         // coordenada y1
   const int     x2,         // coordenada x2
   const int     y2          // coordenada y1
   )
```

Parámetros

chart

[in] Identificador del gráfico.

name

[in] Nombre.

subwin

[in] Número de subventana.

x1

[in] Coordenada X1.

y1

[in] Coordenada Y1.

x2

[in] Coordenada X2.

y2

[in] Coordenada Y2.

© 2000-2025, MetaQuotes Ltd.

Destroy

Elimina un gráfico del gráfico de precios y borra el recurso gráfico.

```
void  Destroy()
```

© 2000-2025, MetaQuotes Ltd.

Update

Representa en la pantalla los cambios realizados.

```
void  Update(
   const bool  redraw=true      // bandera
   )
```

Parámetros

redraw=true

[in] Valor de la bandera

© 2000-2025, MetaQuotes Ltd.

ChartObjectName

Obtiene el nombre del objeto ligado al gráfico

```
string  ChartObjectName()
```

Valor devuelto

Nombre del objeto ligado al gráfico

© 2000-2025, MetaQuotes Ltd.

ResourceName

Obtiene el nombre del recurso gráfico.

```
string  ResourceName()
```

Valor devuelto

Nombre del recurso gráfico.

© 2000-2025, MetaQuotes Ltd.

XAxis

Devuelve el puntero al eje X.

```
CAxis  *XAxis()
```

Valor devuelto

Puntero al eje X.

© 2000-2025, MetaQuotes Ltd.

YAxis

Devuelve el puntero al eje Y.

```
CAxis  *YAxis()
```

Valor devuelto

Puntero al eje Y.

© 2000-2025, MetaQuotes Ltd.

GapSize (método Get)

Devuelve el tamaño de los márgenes entre los elementos del gráfico.

```
int  GapSize()
```

Valor devuelto

Tamaño del margen en píxeles.

GapSize (método Set)

Establece el tamaño de los márgenes entre los elementos del gráfico.

```
void  GapSize(
   const int  size     // tamaño del margen
   )
```

Parámetros

size

[in] Tamaño del margen en píxeles.

© 2000-2025, MetaQuotes Ltd.

BackgroundColor (método Get)

Devuelve el color del fondo.

```
color  BackgroundColor()
```

BackgroundColor (método Set)

Establece el color del fondo.

```
void  BackgroundColor(
   const color  clr      // color del fondo
   )
```

Parámetros

clr

[in] Color del fondo.

© 2000-2025, MetaQuotes Ltd.

BackgroundMain (método Get)

Devuelve el texto del encabezamiento del gráfico

```
string  BackgroundMain()
```

BackgroundMain (método Set)

Establece el texto del encabezamiento del gráfico.

```
void  BackgroundMain(
   const string  main      // texto del encabezamiento
   )
```

Parámetros

main

[in] Texto del encabezamiento del gráfico

© 2000-2025, MetaQuotes Ltd.

BackgroundMainSize (método Get)

Devuelve el tamaño del encabezamiento.

```
int  BackgroundMainSize()
```

Valor devuelto

Tamaño de la fuente del encabezamiento.

BackgroundMainSize (método Set)

Establece el tamaño del encabezamiento.

```
void  BackgroundMainSize(
   const int  size     // tamaño del encabezamiento
   )
```

Parámetros

size

[in] Tamaño de la fuente del encabezamiento.

© 2000-2025, MetaQuotes Ltd.

BackgroundMainColor (método Get)

Devuelve el color del encabezamiento.

```
color  BackgroundMainColor()
```

Valor devuelto

Color del encabezamiento.

BackgroundMainColor (método Set)

Establece el color del encabezamiento.

```
void  BackgroundMainColor(
   const color  clr      // color del encabezamiento
   )
```

Parámetros

clr

[in] Color del encabezamiento.

© 2000-2025, MetaQuotes Ltd.

BackgroundSub (método Get)

Devuelve el texto del subtítulo.

```
string  BackgroundSub()
```

Valor devuelto

Texto del subtítulo.

BackgroundSub (método Set)

Establece el texto del subtítulo.

```
void  BackgroundSub (método Set)(
   const string  sub      // Texto del subtítulo
   )
```

Parámetros

sub

[in] Texto del subtítulo.

© 2000-2025, MetaQuotes Ltd.

BackgroundSubSize (método Get)

Devuelve el tamaño de la fuente para el subtítulo.

```
int BackgroundSubSize()
```

BackgroundSubSize (método Get)

Establece el tamaño del subtítulo.

```
void  BackgroundSubSize(
   const int  size      // tamaño de la fuente para el subtítulo
   )
```

Parámetros

size

[in] Tamaño de la fuente para el subtítulo

© 2000-2025, MetaQuotes Ltd.

BackgroundSubColor (método Get)

Devuelve el color del subtítulo.

```
color  BackgroundSubColor()
```

BackgroundSubColor (método Set)

Establece el color del subtítulo.

```
void  BackgroundSubColor(
   const color  clr      // color del subtítulo
   )
```

Parámetros

clr

[in] Color del subtítulo.

© 2000-2025, MetaQuotes Ltd.

GridLineColor (método Get)

Devuelve el color de las líneas de la cuadrícula

```
color  GridLineColor()
```

Valor devuelto

Color de las líneas de la cuadrícula.

GridLineColor (método Set)

Establece el color de las líneas de la cuadrícula.

```
void  GridLineColor(
   const color  clr      // color de las líneas
   )
```

Parámetros

clr

[in] Color de las líneas de la cuadrícula.

© 2000-2025, MetaQuotes Ltd.

GridBackgroundColor (método Get)

Devuelve el color del fondo de la cuadrícula

```
color  GridBackgroundColor()
```

Valor devuelto

Color del fondo de la cuadrícula

GridBackgroundColor (método Set)

Establece el color del fondo posterior de la cuadrícula

```
void  GridBackgroundColor(
   const color  clr      // color del fondo de la cuadrícula
   )
```

Parámetros

clr

[in] Color del fondo de la cuadrícula

© 2000-2025, MetaQuotes Ltd.

GridCircleRadius (método Get)

Devuelve el radio de los puntos en los nodos de la cuadrícula.

```
int  GridCircleRadius()
```

Valor devuelto

Radio de los puntos en píxeles.

GridCircleRadius (método Set)

Establece el radio de los puntos en los nodos de la cuadrícula

```
void  GridCircleRadius(
   const int  r      // radio
   )
```

Parámetros

r

[in] Radio de los puntos en píxeles.

© 2000-2025, MetaQuotes Ltd.

GridCircleColor (método Get)

Devuelve el color de los puntos en los nodos de la cuadrícula.

```
color  GridCircleColor()
```

Valor devuelto

Color de los puntos.

GridCircleColor (método Set)

Establece el color de los puntos en los nodos de la cuadrícula.

```
void  GridCircleColor(
   const color  clr      // color de los puntos
   )
```

Parámetros

clr

[in] Color de los puntos.

© 2000-2025, MetaQuotes Ltd.

GridHasCircle (método Get)

Devuelve la bandera que indica si hay que dibujar puntos en los nodos de la cuadrícula.

```
bool  GridHasCircle()
```

Valor devuelto

Valor de la bandera.

Nota

true — dibujar puntos

false — no dibujar puntos

GridHasCircle (método Set)

Establece la bandera que indica si hay que dibujar puntos en los nodos de la cuadrícula.

```
void  GridHasCircle(
   const bool  has
   )
```

Parámetros

has

[in] Valor de la bandera.

Nota

true — dibujar puntos

false — no dibujar puntos

© 2000-2025, MetaQuotes Ltd.

GridAxisLineColor (Método Get)

Retorna el valor del color de los ejes reales del gráfico.

```
uint  GridAxisLineColor()
```

Valor devuelto

Color de los ejes reales del gráfico.

GridAxisLineColor (Método Set)

Establece el valor del color de los ejes reales del gráfico.

```
void  GridAxisLineColor(
   const uint  clr      // color de los ejes del gráfico
   )
```

Parámetros

clr

[in] Color de los ejes reales del gráfico.

© 2000-2025, MetaQuotes Ltd.

HistoryNameWidth (método Get)

Devuelve la longitud máxima permitida para representar el nombre de la curva.

```
int  HistoryNameWidth()
```

Valor devuelto

Longitud máxima en píxeles.

Nota

Si el nombre de la curva supera la longitud máxima permitida, entonces se verá cortado y finalizado con puntos supensivos.

HistoryNameWidth (método Set)

Establece la longitud máxima permitida para representar el nombre de la curva.

```
void  HistoryNameWidth(
   const int  width      // longitud máxima
   )
```

Parámetros

width

[in] Longitud máxima en píxeles.

Nota

Si el nombre de la curva supera la longitud máxima permitida, entonces se verá cortado y finalizado con puntos supensivos.

© 2000-2025, MetaQuotes Ltd.

HistoryNameSize (método Get)

Devuelve el tamaño de la fuente para el nombre de la curva.

```
int  HistoryNameSize()
```

Valor devuelto

Tamaño de la fuente para el nombre de la curva.

HistoryNameSize (método Set)

Establece el tamaño de la fuente para el nombre de la curva.

```
void  HistoryNameSize (método Set)(
   const int  size      // tamaño de la fuente para el nombre
   )
```

Parámetros

size

[in] Tamaño de la fuente para el nombre.

© 2000-2025, MetaQuotes Ltd.

HistorySymbolSize (métodos Get)

Devuelve el tamaño de los símbolos de los signos convencionales del gráfico

```
int  HistorySymbolSize()
```

Valor devuelto

Tamaño de los símbolos de los signos convencionales

HistorySymbolSize (método Set)

Establece el tamaño de los símbolos de los signos convencionales del gráfico

```
void  HistorySymbolSize(
   const int  size      // tamaño del símbolo
   )
```

Parámetros

size

[in] Tamaño de los símbolos de los signos convencionales.

© 2000-2025, MetaQuotes Ltd.

TextAdd

Añade texto al gráfico.

Versión para trabajar con la pareja de coordenadas X e Y

```
void  TextAdd(
   const int     x,              // coordenada X
   const int     y,              // coordenada Y
   const string  text,           // texto
   const uint    clr,            // color
   const uint    alignment=0     // alineación
   )
```

Versión para CPoint

```
void  TextAdd(
   const CPoint  &point,         // coordenadas del punto
   const string  text,           // texto
   const uint    clr,            // color
   const uint    alignment=0     // alineación
   )
```

Parámetros

x

[in] Coordenada X.

y

[in] Coordenada Y.

&point

[in] Coordenada del punto.

text

[in] Texto.

clr

[in] Color.

alignment=0

[in] Alineación.

© 2000-2025, MetaQuotes Ltd.

LineAdd

Añade una línea al gráfico.

Versión para trabajar con parejas de coordenadas X e Y

```
void  LineAdd(
   const int   x1,         // coordenada x1
   const int   y1,         // coordenada y1
   const int   x2,         // coordenada x2
   const int   y2,         // coordenada y2
   const uint  clr,        // color
   const uint  style       // estilo
   )
```

Versión para CPoint

```
void  LineAdd2(
   const CPoint  &point1,     // coordenada para el primer punto
   const CPoint  &point2,     // coordenada para el segundo punto
   const uint    clr,         // color
   const uint    style        // estilo
   )
```

Parámetros

x1

[in] Coordenada X1.

y1

[in] Coordenada Y1.

x2

[in] Coordenada X2.

y2

[in] Coordenada Y2.

&point1

[in] coordenada para el primer punto.

&point2

[in] coordenada para el segundo punto.

clr

[in] Color.

style

[in] Estilo.

© 2000-2025, MetaQuotes Ltd.

CurveAdd

Crea una nueva curva y la añade al gráfico.

Versión para trabajar con la coordenada Y (el color de la curva se indica de forma automática)

```
CCurve* CurveAdd(
   const double      &y[],          // coordenadas Y
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nombre de la curva
   )
```

Nota

Para esta curva, los índices de la matriz Y actuarán como coordenadas X.

Versión para trabajar con la pareja de coordenadas X e Y (el color de la curva se indica de forma automática)

```
CCurve*  CurveAdd(
   const double      &x[],          // coordenadas X
   const double      &y[],          // coordenadas Y
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nombre de la curva
   )
```

Versión para trabajar con puntos CPoint2D (el color de la curva se indica de forma automática)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // coordenadas de los puntos
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nombre de la curva
   )
```

Versión para trabajar con el puntero a la función CurveFunction (el color de la curva se indica de forma automática)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // puntero a la función
   const double      from,          // valor inicial del argumento
   const double      to,            // valor final del argumento
   const double      step,          // incremento del argumento
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nombre de la curva
   )
```

© 2000-2025, MetaQuotes Ltd.

Versión para trabajar con la coordenada Y (el usuario indica el color de la curva)

```
CCurve*  CurveAdd(
   const double      &y[],          // coordenadas Y
   const uint        clr,           // color de la curva
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nombre de la curva
   )
```

Nota

Para esta curva, los índices de la matriz Y actuarán como coordenadas X.

Versión para trabajar con la pareja de coordenadas X e Y (el usuario indica el color de la curva)

```
CCurve*  CurveAdd(
   const double      &x[],          // coordenadas X
   const double      &y[],          // coordenadas Y
   const uint        clr,           // color de la curva
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nombre de la curva
   )
```

Versión para trabajar con puntos CPoint2D (el usuario indica el color de la curva)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // coordenadas de los puntos
   const uint        clr,           // color de la curva
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nombre de la curva
   )
```

Versión para trabajar con el puntero a la función CurveFunction (el usuario indica el color de la curva)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // puntero a la función
   const double      from,          // valor inicial del argumento
   const double      to,            // valor final del argumento
   const double      step,          // incremento del argumento
   const uint        clr,           // color de la curva
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nombre de la curva
   )
```

© 2000-2025, MetaQuotes Ltd.

Parámetros

&x[]

[in] Coordenada X.

&y[]

[in] Coordenada Y.

&points[]

[in] Coordenadas de los puntos.

function

[in] Puntero a la función.

from

[in] Valor inicial del argumento.

to

[in] Valor final del argumento.

step

[in] Incremento del argumento.

type

[in] Tipo de curva.

name=NULL

[in] Nombre de la curva.

clr

[in] Color de la curva.

Valor devuelto

Puntero a la curva creada.

© 2000-2025, MetaQuotes Ltd.

CurvePlot

Representa una curva creada con anterioridad con el índice establecido.

```
bool  CurvePlot(
   const int  index      // índice
   )
```

Parámetros

index

[in] Índice de la curva

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CurvePlotAll

Representa todas las curvas añadidas al gráfico con anterioridad.

```
bool  CurvePlotAll()
```

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CurveGetByIndex

Obtiene una curva según el índice establecido.

```
CCurve*  CurveGetByIndex(
   const int  index      // índice de la curva
   )
```

Parámetros

index

[in] Índice de la curva.

Valor devuelto

Puntero a la curva con el índice establecido.

© 2000-2025, MetaQuotes Ltd.

CurveGetByName

Obtiene una curva según el nombre establecido.

```
CCurve*  CurveGetByName(
   const string  name      // nombre de la curva
   )
```

Parámetros

name

[in] Nombre de la curva.

Valor devuelto

Puntero a la primera curva encontrada con el nombre establecido.

© 2000-2025, MetaQuotes Ltd.

CurveRemoveByName

Elimina una curva según el nombre establecido.

```
bool  CurveRemoveByName(
   const string  name      // nombre de la curva
   )
```

Parámetros

name

[in] Nombre de la curva que se debe eliminar.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CurveRemoveByIndex

Elimina una curva según el índice establecido.

```
bool  CurveRemoveByIndex(
   const int  index      // índice de la curva
   )
```

Parámetros

index

[in] Índice de la curva que se debe eliminar.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

CurvesTotal

Retorna el número de curvas para el gráfico dado.

```
int  CurvesTotal()
```

Valor devuelto

Número de curvas.

Nota

Se consideran todas las curvas pertenecientes a este gráfico, independientemente de su estilo de dibujado y su visibilidad.

© 2000-2025, MetaQuotes Ltd.

MarksToAxisAdd

Añade una marca de la escala ("muesca") al eje indicado del gráfico.

```
bool  MarksToAxisAdd(
   const double         &marks[],        // coordenadas de la "muesca"
   const int            mark_size,       // tamaño de las "muescas"
   ENUM_MARK_POSITION   position,        // ubicación de las "muescas"
   const int            dimension=0      // medición
   )
```

Parámetros

&marks[]

[in] Coordenadas de las "muescas"

mark_size

[in] Tamaño de las "muescas"

position

[in] Ubicación de las "muescas"

dimension=0

[in] 0 — adición al eje X,

1 — adición al eje Y

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

MajorMarkSize (método Get)

Devuelve el tamaño de las "muescas" de la escala en los ejes de coordenadas.

```
int  MajorMarkSize()
```

MajorMarkSize (método Set)

Establece el tamaño de las "muescas" de la escala en los ejes de coordenadas.

```
void  MajorMarkSize(
   const int  size      // tamaño de las "muescas"
   )
```

Parámetros

size

[in] Tamaño de las "muescas" en píxeles.

© 2000-2025, MetaQuotes Ltd.

FontSet

Establece los parámetros de la fuente actual.

```
bool  FontSet(
   const string  name,         // nombre
   const int     size,         // tamaño
   const uint    flags=0,      // banderas
   const uint    angle=0       // ángulo
   )
```

Parámetros

name

[in] Nombre.

size

[in] Tamaño.

flags=0

[in] Banderas.

angle=0

[in] Ángulo.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

FontGet

Obtiene los parámetros de la fuente actual.

```
void  FontGet(
   string  &name,      // nombre
   int     &size,      // tamaño
   uint    &flags,     // banderas
   uint    &angle      // ángulo
   )
```

Parámetros

&name

[out] Nombre.

&size

[out] Tamaño.

&flags

[out] Banderas.

&angle

[out] Ángulo.

© 2000-2025, MetaQuotes Ltd.

Attach

Versión para obtener del objeto OBJ_BITMAP_LABEL el recurso gráfico y su vinculación al ejemplar de la clase CGraphic:

```
bool  Attach(
   const long     chart_id,      // identificador del gráfico
   const string   objname        // nombre del objeto gráfico
   )
```

Versión para crear un recurso gráfico para el objeto OBJ_BITMAP_LABEL y vincularlo a un ejemplar de la clase CGraphic:

```
bool  Attach(
   const long     chart_id,      // identificador del gráfico
   const string   objname,       // nombre del objeto gráfico
   const int      width,         // anchura de la imagen
   const int      height         // altura de la imagen
   )
```

Parámetros

chart_id

[in] Identificador del gráfico.

objname

[in] Denominación (nombre) del objeto gráfico.

width

[in] Anchura de la imagen en el recurso.

height

[in] Altura de la imagen en el recurso.

Valor devuelto

true — en caso de éxito, false — si no se ha logrado vincular el objeto.

© 2000-2025, MetaQuotes Ltd.

CalculateMaxMinValues

Realiza el cálculo (recálculo) de los valores máximos y mínimos del gráfico en los dos ejes.

```
void  CalculateMaxMinValues()
```

© 2000-2025, MetaQuotes Ltd.

Height

Retorna la altura del gráfico en píxeles.

```
int  Height()
```

Valor devuelto

Altura del gráfico en píxeles.

© 2000-2025, MetaQuotes Ltd.

IndentDown (Método Get)

Retorna el margen del gráfico con respecto al límite inferior.

```
int  IndentDown()
```

Valor devuelto

Tamaño del margen en píxeles.

IndentDown (Método Set)

Establece el margen del gráfico con respecto al límite inferior.

```
void  IndentDown(
   const int  down      // tamaño del margen
   )
```

Parámetros

down

[in] Tamaño del margen en píxeles.

© 2000-2025, MetaQuotes Ltd.

IndentLeft (Método Get)

Retorna el margen del gráfico con respecto al límite izquierdo.

```
int  IndentLeft()
```

Valor devuelto

Tamaño del margen en píxeles.

IndentLeft (Método Set)

Establece el margen del gráfico con respecto al límite izquierdo.

```
void  IndentLeft(
   const int  left      // tamaño del margen
   )
```

Parámetros

left

[in] Tamaño del margen en píxeles.

© 2000-2025, MetaQuotes Ltd.

IndentRight (Método Get)

Retorna el margen del gráfico con respecto al límite derecho.

```
int  IndentRight()
```

Valor devuelto

Tamaño del margen en píxeles.

IndentRight (Método Set)

Establece el margen del gráfico con respecto al límite derecho.

```
void  IndentRight(
   const int  right      // tamaño del margen
   )
```

Parámetros

right

[in] Tamaño del margen en píxeles.

© 2000-2025, MetaQuotes Ltd.

IndentUp (Método Get)

Retorna el margen del gráfico con respecto al límite superior.

```
int  IndentUp()
```

Valor devuelto

Tamaño del margen en píxeles.

IndentUp (Método Set)

Establece el margen del gráfico con respecto al límite superior.

```
void  IndentUp(
   const int  up      // tamaño del margen
   )
```

Parámetros

up

[in] Valor del margen en píxeles.

© 2000-2025, MetaQuotes Ltd.

Redraw

Redibuja el gráfico.

```
bool  Redraw(
   const bool  rescale=false      // valor de la bandera
   )
```

Parámetros

rescale=false

[in] bandera que indica si hay que aplicar el reescalado del gráfico.

Valor devuelto

true en caso de éxito, de lo contrario, false.

© 2000-2025, MetaQuotes Ltd.

ResetParameters

Resetea los parámetros necesarios para redibujar el gráfico.

```
void  ResetParameters()
```

© 2000-2025, MetaQuotes Ltd.

ScaleX

Escala el valor en el eje X.

```
virtual int  ScaleX(
   double  x      // valor del eje X
   )
```

Parámetros

x

[in] Valor real del eje X.

Valor devuelto

Valor en píxeles.

Nota

Escala el valor real en píxeles para la representación en el gráfico.

© 2000-2025, MetaQuotes Ltd.

ScaleY

Escala el valor en el eje Y.

```
virtual int  ScaleY(
   double  y      // valor del eje y
   )
```

Parámetros

y

[in] Valor real del eje y.

Valor devuelto

Valor en píxeles.

Nota

Escala el valor real en píxeles para la representación en el gráfico.

© 2000-2025, MetaQuotes Ltd.

SetDefaultParameters

Da a los parámetros del gráfico los valores por defecto.

```
void  SetDefaultParameters()
```

© 2000-2025, MetaQuotes Ltd.

Width

Retorna la anchura del gráfico en píxeles.

```
int  Width()
```

Valor devuelto

Anchura del gráfico en píxeles.

© 2000-2025, MetaQuotes Ltd.

Indicadores técnicos y Series temporales

Esta sección contiene detalles técnicos de las clases de los indicadores técnicos y de las series temporales, así como las descripciones de los componentes correspondientes de la Librería Estándar de MQL5.

Las clases de los indicadores técnicos y de las series temporales le ayudarán a ahorrar tiempo en el momento de desarrollar sus propias aplicaciones (guiones y asesores expertos).

Estas clases de la Librería Estándar MQL5 se encuentran en el directorio de trabajo del terminal Include\Indicators.

Clase/grupo	Descripción
Clases base	Grupo de clases base y auxiliares
Clases de las series temporales	Grupo de clases de las series temporales
Indicadores de tendencia	Grupo de clases indicadoras de tendencia
Osciladores	Grupo de clases de los osciladores
Indicadores de volumen	Grupo de las clases indicadoras de volumen
Indicadores Bill Williams	Clases del indicador Bill Williams
Indicadores personalizados	Clase del indicador personalizado

© 2000-2025, MetaQuotes Ltd.

Clases base, indicador técnico auxiliar y series temporales

Esta sección contiene los detalles técnicos de las clases base, indicador técnico auxiliar y series temporales, así como la descripción de los componentes correspondientes de la Librería Estándar MQL5.

Clase/grupo	Descripción
CSpreadBuffer	Clase del búfer del historial del spread
CTimeBuffer	Clase del búfer del historial de los precios de apertura
CTickVolumeBuffer	Clase del búfer del historial de los ticks de los volúmenes
CRealVolumeBuffer	Clase del búfer del historial de los volúmenes reales
CDoubleBuffer	Clase base del búfer de los datos de tipo double
COpenBuffer	Clase del búfer de los precios de apertura
CHighBuffer	Clase del búfer de los precios máximos (High)
CLowBuffer	Clase del búfer de los precios mínimos (Low)
CCloseBuffer	Clase del búfer de los precios de cierre
CIndicatorBuffer	Clase del búfer del indicador técnico
CSeries	Clase base para acceder a los datos de las series temporales
CPriceSeries	Clase base para acceder a los datos de los precios
CIndicator	Clase base del indicador técnico
CIndicators	Colección del indicador técnico y las series

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

CSpreadBuffer

La clase CSpreadBuffer facilita el acceso al historial de spreads de las barras.

Descripción

La clase CSpreadBuffer proporciona acceso al historial de spreads de las barras.

Declaración

```
class CSpreadBuffer: public CArrayInt
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayInt
      CSpreadBuffer

Métodos de la clase

Atributos	
Size	Establece el tamaño del búfer
Configuraciones	
SetSymbolPeriod	Establece el símbolo y el período
Métodos de acceso a los datos	
At	Obtiene el elemento del búfer por índice
Métodos de actualización de datos	
virtual Refresh	Actualiza el búfer
virtual RefreshCurrent	Actualiza el valor actual

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayInt

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

Establece el tamaño del búfer.

```
void  Size(
   const int  size      // tamaño nuevo
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Establece el símbolo y el período.

```
void  SetSymbolPeriod(
   const string           symbol,     // símbolo
   const ENUM_TIMEFRAMES  period      // período
   )
```

Parámetros

symbol

[in] Nuevo símbolo.

period

[in] Nuevo período (enumeración ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del búfer por su índice.

```
int  At(
   const int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento del búfer.

Valor devuelto

Elemento del búfer con el índice especificado.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza el búfer.

```
virtual bool  Refresh()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Actualiza el elemento (cero) actual del búfer.

```
virtual bool  RefreshCurrent()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

CTimeBuffer

La clase CTimeBuffer facilita el acceso al historial de las horas de apertura de las barras.

Descripción

La clase CTimeBuffer proporciona acceso al historial de las horas de apertura de las barras.

Declaración

```
class CTimeBuffer: public CArrayLong
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayLong
      CTimeBuffer

Métodos de la clase

Atributos	
Size	Establece el tamaño del búfer
Configuraciones	
SetSymbolPeriod	Establece el símbolo y el período
Métodos de acceso a los datos	
At	Obtiene el elemento del búfer por índice
Métodos de actualización de datos	
virtual Refresh	Actualiza el búfer
virtual RefreshCurrent	Actualiza el valor actual

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

Establece el tamaño del búfer.

```
void  Size(
   const int  size      // tamaño nuevo
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Establece el símbolo y el período.

```
void  SetSymbolPeriod(
   const string           symbol,     // símbolo
   const ENUM_TIMEFRAMES  period      // período
   )
```

Parámetros

symbol

[in] Nuevo símbolo.

period

[in] Nuevo período (enumeración ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del búfer por su índice.

```
long  At(
   const int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento del búfer.

Valor devuelto

Elemento del búfer con el índice especificado.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza el búfer.

```
virtual bool  Refresh()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Actualiza el elemento (cero) actual del búfer.

```
virtual bool  RefreshCurrent()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

CTickVolumeBuffer

La clase CTickVolumeBuffer facilita el acceso al historial de los volúmenes de los ticks de las barras.

Descripción

La clase CTickVolumeBuffer proporciona acceso al historial de los volúmenes de los ticks de las barras.

Declaración

```
class CTickVolumeBuffer: public CArrayLong
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayLong
      CTickVolumeBuffer

Métodos de la clase

Atributos	
Size	Establece el tamaño del búfer
Configuraciones	
SetSymbolPeriod	Establece el símbolo y el período
Métodos de acceso a los datos	
At	Obtiene el elemento del búfer por índice
Métodos de actualización de datos	
virtual Refresh	Actualiza el búfer
virtual RefreshCurrent	Actualiza el valor actual

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

Establece el tamaño del búfer.

```
void  Size(
   const int  size      // tamaño nuevo
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Establece el símbolo y el período.

```
void  SetSymbolPeriod(
   const string           symbol,     // símbolo
   const ENUM_TIMEFRAMES  period      // período
   )
```

Parámetros

symbol

[in] Nuevo símbolo.

period

[in] Nuevo período (enumeración ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del búfer por su índice.

```
long  At(
   const int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento del búfer.

Valor devuelto

Elemento del búfer con el índice especificado.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza el búfer.

```
virtual bool  Refresh()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Actualiza el elemento (cero) actual del búfer.

```
virtual bool  RefreshCurrent()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

CRealVolumeBuffer

La clase CRealVolumeBuffer facilita el acceso al historial de los volúmenes reales de las barras.

Descripción

La clase CTickVolumeBuffer proporciona acceso al historial de los volúmenes reales de las barras.

Declaración

```
class CRealVolumeBuffer: public CArrayLong
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayLong
      CRealVolumeBuffer

Métodos de la clase

Atributos	
Size	Establece el tamaño del búfer
Configuraciones	
SetSymbolPeriod	Establece el símbolo y el período
Métodos de acceso a los datos	
At	Obtiene el elemento del búfer por índice
Métodos de actualización de datos	
virtual Refresh	Actualiza el búfer
virtual RefreshCurrent	Actualiza el valor actual

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

Establece el tamaño del búfer.

```
void  Size(
   const int  size      // tamaño nuevo
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Establece el símbolo y el período.

```
void  SetSymbolPeriod(
   const string           symbol,     // símbolo
   const ENUM_TIMEFRAMES  period      // período
   )
```

Parámetros

symbol

[in] Nuevo símbolo.

period

[in] Nuevo período (enumeración ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del búfer por su índice.

```
long  At(
   const int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento del búfer.

Valor devuelto

Elemento del búfer con el índice especificado.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza el búfer.

```
virtual bool  Refresh()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Actualiza el elemento (cero) actual del búfer.

```
virtual bool  RefreshCurrent()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

CDoubleBuffer

La clase base CDoubleBuffer facilita el acceso a los búferes de datos de tipo double.

Descripción

La clase CDoubleBuffer proporciona acceso a los búferes de datos de tipo double.

Declaración

```
class CDoubleBuffer: public CArrayDouble
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayDouble
      CDoubleBuffer

Descendientes directos

CCloseBuffer, CHighBuffer, CIndicatorBuffer, CLowBuffer, COpenBuffer

Métodos de la clase

Atributos	
Size	Establece el tamaño del búfer
Configuraciones	
SetSymbolPeriod	Establece el símbolo y el período
Métodos de acceso a los datos	
At	Obtiene el elemento del búfer
Métodos de actualización de datos	
virtual Refresh	Actualiza el búfer
virtual RefreshCurrent	Actualiza el valor actual

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

Establece el tamaño del búfer.

```
void  Size(
   const int  size      // tamaño nuevo
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Establece el símbolo y el período.

```
void  SetSymbolPeriod(
   const string           symbol,     // símbolo
   const ENUM_TIMEFRAMES  period      // período
   )
```

Parámetros

symbol

[in] Nuevo símbolo.

period

[in] Nuevo período (enumeración ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del búfer por su índice.

```
double  At(
   const int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento del búfer.

Valor devuelto

Elemento del búfer con el índice especificado.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza el búfer.

```
virtual bool  Refresh()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Actualiza el elemento (cero) actual del búfer.

```
virtual bool  RefreshCurrent()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

COpenBuffer

La clase COpenBuffer facilita el acceso al historial de los precios de apertura de las barras.

Descripción

La clase COpenBuffer proporciona acceso al historial de los precios de apertura de las barras.

Declaración

```
class COpenBuffer: public CDoubleBuffer
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        COpenBuffer

Métodos de la clase

Métodos de actualización de datos	
virtual Refresh	Actualiza el búfer
virtual RefreshCurrent	Actualiza el valor actual

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Métodos heredados de la clase CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza el búfer.

```
virtual bool  Refresh()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Actualiza el elemento (cero) actual del búfer.

```
virtual bool  RefreshCurrent()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

CHighBuffer

La clase CHighBuffer facilita el acceso al historial de los precios máximos de las barras.

Descripción

La clase CHighBuffer proporciona acceso al historial de los precios máximos de las barras.

Declaración

```
class CHighBuffer: public CDoubleBuffer
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CHighBuffer

Métodos de la clase

Métodos de actualización de datos	
virtual Refresh	Actualiza el búfer
virtual RefreshCurrent	Actualiza el valor actual

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Métodos heredados de la clase CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza el búfer.

```
virtual bool  Refresh()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Actualiza el elemento (cero) actual del búfer.

```
virtual bool  RefreshCurrent()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

CLowBuffer

La clase CLowBuffer facilita el acceso al historial de los precios mínimos de las barras.

Descripción

La clase CLowBuffer proporciona acceso al historial de los precios mínimos de las barras.

Declaración

```
class CLowBuffer: public CDoubleBuffer
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CLowBuffer

Métodos de la clase

Métodos de actualización de datos	
virtual Refresh	Actualiza el búfer
virtual RefreshCurrent	Actualiza el valor actual

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Métodos heredados de la clase CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza el búfer.

```
virtual bool  Refresh()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Actualiza el elemento (cero) actual del búfer.

```
virtual bool  RefreshCurrent()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

CCloseBuffer

La clase CCloseBuffer facilita el acceso al historial de los precios de cierre de las barras.

Descripción

La clase CCloseBuffer proporciona acceso al historial de los precios de cierre de las barras.

Declaración

```
class CCloseBuffer: public CDoubleBuffer
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CCloseBuffer

Métodos de la clase

Métodos de actualización de datos	
virtual Refresh	Actualiza el búfer
virtual RefreshCurrent	Actualiza el valor actual

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Métodos heredados de la clase CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza el búfer.

```
virtual bool  Refresh()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Actualiza el elemento (cero) actual del búfer.

```
virtual bool  RefreshCurrent()
```

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

CIndicatorBuffer

La clase CIndicatorBuffer facilita el acceso a loas datos del búfer del indicador.

Descripción

La clase CIndicatorBuffer facilita el acceso a loas datos del búfer del indicador técnico.

Declaración

```
class CIndicatorBuffer: public CDoubleBuffer
```

Título

```
#include <Indicators\Indicator.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayDouble
CDoubleBuffer
CIndicatorBuffer

Métodos de la clase

Atributos	
Offset	Obtiene/Establece el offset del búfer
Name	Obtiene/Establece el nombre del búfer
Métodos de acceso a los datos	
At	Obtiene el elemento del búfer
Métodos de actualización de datos	
Refresh	Actualiza el búfer
RefreshCurrent	Actualiza solamente el valor actual

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Métodos heredados de la clase CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Offset

Obtiene el offset del búfer

```
int  Offset() const
```

Valor devuelto

Offset del búfer.

Offset

Establece el offset del búfer.

```
void  Offset(
   const int  offset      // offset
   )
```

Parámetros

offset

[in] Nuevo offset del búfer.

© 2000-2025, MetaQuotes Ltd.

Name

Obtiene el nombre del búfer.

```
string  Name() const
```

Valor devuelto

Nombre del búfer.

Name

Establece el nombre del búfer.

```
void  Name(
   const string  name      // nombre
   )
```

Parámetros

name

[in] Nombre nuevo del búfer.

© 2000-2025, MetaQuotes Ltd.

At

Obtiene el elemento del búfer por su índice.

```
double  At(
   int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento del búfer.

Valor devuelto

Elemento del búfer con el índice especificado.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza el búfer completo.

```
bool  Refresh(
   const int  handle,     // manejador
   const int  num         // número de búfer
   )
```

Parámetros

handle

[in] Manejador del indicador.

num

[in] Índice del búfer del indicador.

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Actualiza el elemento actual (cero) del búfer.

```
bool  RefreshCurrent(
   const int  handle,     // manejador del indicador
   const int  num         // número de búfer
   )
```

Parámetros

handle

[in] Manejador del indicador.

num

[in] Número del búfer.

Valor devuelto

true - si se ejecuta correctamente, false - si el búfer no se puede actualizar.

© 2000-2025, MetaQuotes Ltd.

CSeries

CSeries es la clase base de la Librería Estándar para acceder a los datos de las series temporales.

Descripción

La clase CSeries facilita a sus descendientes el acceso a las funciones MQL5 de series temporales.

Declaración

```
class CSeries: public CArrayObj
```

Título

```
#include <Indicators\Series.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries

Descendientes directos

CIndicator, CiRealVolume, CiSpread, CiTickVolume, CiTime, CPriceSeries

Métodos de la clase

Atributos	
Name	Obtiene el nombre de la serie temporal o del indicador
BuffersTotal	Obtiene el número de búferes de la serie temporal o del indicador
Timeframe	Obtiene la bandera del periodo de tiempo de la serie temporal o del indicador
Symbol	Obtiene el símbolo de la serie temporal o del indicador
Period	Obtiene el período de la serie temporal o del indicador
RefreshCurrent	Obtiene/Establece la bandera para actualizar los datos actuales
Métodos de acceso a los datos	
virtual BufferResize	Establece el tamaño del búfer de la serie temporal o del indicador
Métodos de actualización de datos	
virtual Refresh	Actualiza los datos de la serie temporal o del indicador

© 2000-2025, MetaQuotes Ltd.

Atributos	
PeriodDescription	Obtiene el período en formato string

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

Name

Obtiene el nombre de la serie temporal o del indicador

```
string  Name() const
```

Valor devuelto

El nombre de la serie temporal o del indicador.

© 2000-2025, MetaQuotes Ltd.

BuffersTotal

Obtiene el número de búferes de la serie temporal o del indicador.

```
int  BuffersTotal() const
```

Valor devuelto

El número de búferes de la serie temporal o del indicador.

Nota

La serie temporal solo tiene un búfer.

© 2000-2025, MetaQuotes Ltd.

Timeframe

Obtiene la bandera del periodo de tiempo de la serie temporal o del indicador.

```
int  Timeframe() const
```

Valor devuelto

La bandera del periodo de tiempo de la serie temporal o del indicador

Nota

Es la bandera de visibilidad de algunas series temporales.

© 2000-2025, MetaQuotes Ltd.

Symbol

Obtiene el símbolo de la serie temporal o del indicador.

```
string  Symbol() const
```

Valor devuelto

El símbolo de la serie temporal o del indicador.

© 2000-2025, MetaQuotes Ltd.

Period

Obtiene el periodo de la serie temporal o del indicador.

```
ENUM_TIMEFRAMES  Period() const
```

Valor devuelto

El periodo (valor de la enumeración ENUM_TIMEFRAMES) de la serie temporal o del indicador.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Establece la bandera para actualizar los valores actuales de la serie temporal o del indicador.

```
string  RefreshCurrent(
   const bool  flag      // bandera nueva
   )
```

Parámetros

flag

[in] Bandera nueva.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

BufferSize

Devuelve la cantidad de datos disponibles en el búfer de la serie temporal o del indicador.

```
int  BufferSize() const
```

Valor devuelto

Cantidad de datos disponibles en el búfer de la serie temporal o del indicador.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Establece el tamaño del búfer de la serie temporal o del indicador.

```
virtual bool  BufferResize(
   const int  size      // tamaño nuevo
   )
```

Parámetros

size

[in] Nuevo tamaño de los búferes.

Valor devuelto

true - si se ejecuta correctamente, false - en caso contrario.

Nota

Todos los búferes de la serie temporal o del indicador tienen el mismo tamaño.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza los datos de la serie temporal o del indicador.

```
virtual void  Refresh(
   const int  flags      // banderas
   )
```

Parámetros

flags

[in] Períodos temporales a actualizar (bandera).

© 2000-2025, MetaQuotes Ltd.

PeriodDescription

Obtiene la representación string de la enumeración ENUM_TIMEFRAMES especificada.

```
string  PeriodDescription(
   const int  val=0      // valor
   )
```

Parámetros

val=0

[in] Valor a convertir.

Valor devuelto

La representación string de la enumeración ENUM_TIMEFRAMES.

Nota

Si no se especifica ningún valor, o éste es igual a cero, devuelve el periodo de tiempo de la serie temporal o el indicador.

© 2000-2025, MetaQuotes Ltd.

CPriceSeries

CPriceSeries es la clase base para acceder a los datos del precio.

Descripción

La clase CSeries facilita a sus descendientes el acceso a las funciones MQL5, para trabajar con los datos de los precios.

Declaración

```
class CPriceSeries: public CSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries

Descendientes directos

CiClose, CiHigh, CiLow, CiOpen

Métodos de la clase

Métodos de creación	
virtual BufferResize	Establece el tamaño del búfer
Métodos de acceso a los datos	
virtual GetData	Obtiene el elemento del búfer especificado por su índice
Métodos de actualización de datos	
virtual Refresh	Actualiza los datos de la serie temporal
Métodos de búsqueda de datos	
virtual MinIndex	Obtiene el índice del elemento mínimo en el intervalo especificado
virtual MinValue	Obtiene el valor y el índice del elemento mínimo en el intervalo especificado

© 2000-2025, MetaQuotes Ltd.

Métodos de creación	
virtual MaxIndex	Obtiene el índice del elemento máximo en el intervalo especificado
virtual MaxValue	Obtiene el valor y el índice del elemento máximo en el intervalo especificado

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

BufferResize

Establece el nuevo tamaño del búfer.

```
virtual void  BufferResize(
   const int  size      // tamaño nuevo
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene el elemento del búfer especificado por su índice.

```
double  GetData(
   const int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento del búfer.

Valor devuelto

El elemento del búfer con el índice especificado, o EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Refresh

Almacena los datos de las series temporales

```
virtual void  Refresh(
   const int  flags=OBJ_ALL_PERIODS      // banderas de la serie temporal
   )
```

Parámetros

flags=OBJ_ALL_PERIODS

[in] Períodos temporales a actualizar (bandera).

© 2000-2025, MetaQuotes Ltd.

MinIndex

Obtiene el índice del elemento mínimo en el intervalo especificado.

```
virtual int  MinIndex(
   const int  start,      // índice de inicio
   const int  count       // número de elementos a escanear
   ) const
```

Parámetros

start

[in] Índice de inicio.

count

[in] Número de elementos.

Valor devuelto

El índice del elemento mínimo en el intervalo especificado, o -1 en caso de error.

© 2000-2025, MetaQuotes Ltd.

MinValue

Obtiene el valor y el índice del elemento mínimo en el intervalo especificado.

```
virtual double  MinValue(
   const int    start,      // índice de inicio
   const int    count,      // número de elementos a escanear
   int&         index       // referencia a la variable del índice
   ) const
```

Parámetros

start

[in] Índice de inicio.

count

[in] Número de elementos.

index

[out] Referencia a la variable de tipo entero.

Valor devuelto

El valor del elemento mínimo del búfer en el intervalo especificado, o EMPTY_VALUE en caso de error.

Nota

El índice del elemento mínimo se almacena en la variable index.

© 2000-2025, MetaQuotes Ltd.

MaxIndex

Obtiene el índice del elemento máximo en el intervalo especificado.

```
virtual int  MaxIndex(
   const int  start,      // índice de inicio
   const int  count       // número de elementos a escanear
   ) const
```

Parámetros

start

[in] Índice de inicio.

count

[in] Número de elementos.

Valor devuelto

El índice del elemento máximo en el intervalo especificado, o -1 en caso de error.

© 2000-2025, MetaQuotes Ltd.

MaxValue

Obtiene el valor y el índice del elemento máximo en el intervalo especificado.

```
virtual double  MaxValue(
   const int   start,     // índice de inicio
   const int   count,     // número de elementos a escanear
   int&        index      // referencia a la variable del índice
   ) const
```

Parámetros

start

[in] Índice de inicio.

count

[in] Número de elementos.

index

[out] Referencia a la variable de tipo entero.

Valor devuelto

El valor del elemento máximo del búfer en el intervalo especificado, o EMPTY_VALUE en caso de error.

Nota

El índice del elemento máximo se almacena en la variable index.

© 2000-2025, MetaQuotes Ltd.

CIndicator

CIndicator es la clase base de los indicadores técnicos de la Librería Estándar MQL.

Descripción

La clase CIndicator facilita a todos sus descendientes el acceso a los indicadores técnicos de MQL5.

Declaración

```
class CIndicator: public CSeries
```

Título

```
#include <Indicators\Indicator.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator

Descendientes directos

CiAC, CiAD, CiADX, CiADXWilder, CiAlligator, CiAMA, CiAO, CiATR, CiBands, CiBearsPower, CiBullsPower, CiBWMFI, CiCCI, CiChaikin, CiCustom, CiDEMA, CiDeMarker, CiEnvelopes, CiForce, CiFractals, CiFrAMA, CiGator, CiIchimoku, CiMA, CiMACD, CiMFI, CiMomentum, CiOBV, CiOsMA, CiRSI, CiRVI, CiSAR, CiStdDev, CiStochastic, CiTEMA, CiTriX, CiVIDyA, CiVolumes, CiWPR

Métodos de la clase

Atributos	
Handle	Obtiene el manejador del indicador.
Status	Obtiene el estado del indicador.
FullRelease	Establece la bandera para liberar el manejador del indicador.
Creación	
Create	Crea los indicadores
BufferResize	Establece el nuevo tamaño del búfer
Métodos de acceso a los datos	
GetData	Copia los datos del búfer del indicador
Métodos de actualización de datos	
Refresh	Actualiza los datos del indicador

© 2000-2025, MetaQuotes Ltd.

Atributos	
Encuentra los valores Min/Max	
Minimum	Obtiene el índice del elemento mínimo del búfer especificado en el intervalo dado.
MinValue	Obtiene el valor y el índice del elemento mínimo del búfer especificado en el intervalo dado.
Maximum	Obtiene el índice del elemento máximo del búfer especificado en el intervalo dado.
MaxValue	Obtiene el valor y el índice del elemento máximo del búfer especificado en el intervalo dado.
Conversión de enumeraciones	
MethodDescription	Obtiene en forma de string el valor de la enumeración ENUM_MA_METHOD
PriceDescription	Obtiene en forma de string el valor de la enumeración ENUM_APPLIED_PRICE
VolumeDescription	Obtiene en forma de string el valor de la enumeración ENUM_APPLIED_VOLUME
Trabajar con gráficos	
AddToChart	Añade el indicador al gráfico
DeleteFromChart	Borra el indicador del gráfico

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Clases derivadas:

- CiAC
- CiAD
- CiADX

© 2000-2025, MetaQuotes Ltd.

- CiADXWilder
- CiAlligator
- CiAMA
- CiAO
- CiATR
- CiBands
- CiBearsPower
- CiBullsPower
- CiBWMFI
- CiCCI
- CiChaikin
- CiDEMA
- CiDeMarker
- CiEnvelopes
- CiForce
- CiFractals
- CiFrAMA
- CiGator
- CiIchimoku
- CiMA
- CiMACD
- CiMFI
- CiMomentum
- CiOBV
- CiOsMA
- CiRSI
- CiRVI
- CiSAR
- CiStdDev
- CiStochastic
- CiTEMA
- CiTriX
- CiVIDyA
- CiVolumes
- CiWPR

© 2000-2025, MetaQuotes Ltd.

Handle

Obtiene el manejador del indicador.

```
int  Handle() const
```

Valor devuelto

Manejador del indicador.

© 2000-2025, MetaQuotes Ltd.

Status

Obtiene el estado del indicador.

```
string  Status() const
```

Valor devuelto

Es estado de la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

FullRelease

Establece la bandera para liberar el manejador del indicador.

```
void  FullRelease(
   const bool  flag=true      // bandera
   )
```

Parámetros

flag

[in] Nuevo valor de la bandera que libera el manejador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados.

```
bool  Create(
   const string           symbol,          // símbolo
   const ENUM_TIMEFRAMES  period,          // periodo
   const ENUM_INDICATOR   type,            // tipo
   const int              num_params,      // número de parámetros
   const MqlParam&        params[]         // referencia al array de parámetros
   )
```

Parámetros

symbol

[in] Nombre del símbolo.

period

[in] Periodo (enumeración ENUM_TIMEFRAMES).

type

[in] Tipo del indicador (enumeración ENUM_INDICATOR).

num_params

[in] Número de parámetros del indicador.

params

[in] Referencia al array de parámetros del indicador.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Establece el tamaño del búfer del indicador.

```
virtual bool  BufferResize(
   const int  size      // tamaño
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

Valor devuelto

true - si se ejecuta correctamente, false - en caso contrario.

Nota

Todos los búferes del indicador tienen el mismo tamaño.

© 2000-2025, MetaQuotes Ltd.

BarsCalculated

Devuelve el número de datos calculados por el indicador.

```
int  BarsCalculated() const;
```

Valor devuelto

Devuelve la cantidad de datos calculados en el búfer del indicador, o -1 en caso de error (los datos no se han calculado todavía).

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene el elemento especificado del búfer del indicador. Refresh() debe llamarse para trabajar con datos recientes antes de utilizar el método.

```
double  GetData(
   const int  buffer_num,     // número del búfer
   const int  index           // índice del elemento
   ) const
```

Parámetros

buffer_num

[in] Número del búfer.

index

[in] Índice del elemento.

Valor devuelto

Si se ejecuta correctamente devuelve el valor numérico del elemento, o EMPTY_VALUE en caso de error.

GetData

Obtiene los datos del búfer del indicador por la posición inicial y el número de datos necesarios.

```
int  GetData(
   const int      start_pos,      // posición
   const int      count,          // número de elementos requeridos
   const int      buffer_num,     // número de búfer
   double&        buffer[]        // array de datos destino
   ) const
```

Parámetros

start_pos

[in] Posición inicial del búfer del indicador.

count

[in] Número de elementos requeridos.

buffer_num

[in] Número del búfer del indicador.

buffer

[in] Referencia al array de datos destino.

Valor devuelto

Si se ejecuta correctamente, devuelve el número de elementos recibidos del búfer del indicador especificado; en otro caso -1.

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene los datos del búfer del indicador por hora de inicio y número de datos necesarios.

```
int  GetData(
   const datetime  start_time,     // hora de inicio
   const int       count,          // número de elementos requeridos
   const int       buffer_num,     // número de búfer
   double&         buffer[]        // array de datos destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

count

[in] Número de elementos requeridos.

buffer_num

[in] Número del búfer del indicador.

buffer

[in] Referencia al array destino.

Valor devuelto

Si se ejecuta correctamente, devuelve el número de elementos recibidos del búfer del indicador especificado; en otro caso -1.

GetData

Obtiene los datos del búfer del indicador por hora de inicio y parada, y número de datos necesarios.

```
int  GetData(
   const datetime  start_time,     // hora de inicio
   const datetime  stop_time,      // hora de parada
   const int       buffer_num,     // número de búfer
   double&         buffer[]        // array de datos destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

stop_time

[in] Hora de parada.

buffer_num

[in] Número del búfer del indicador.

buffer

© 2000-2025, MetaQuotes Ltd.

[in] Referencia al array destino.

Valor devuelto

Si se ejecuta correctamente, devuelve el número de elementos recibidos del búfer del indicador especificado; en otro caso -1.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza los datos del indicador. Se recomienda llamar al método antes de utilizar GetData().

```
virtual void  Refresh(
   int  flags=OBJ_ALL_PERIODS      // banderas
   )
```

Parámetros

flags=OBJ_ALL_PERIODS

[in] Banderas de actualización del período de tiempo.

© 2000-2025, MetaQuotes Ltd.

Minimum

Devuelve el índice del elemento mínimo del búfer en el intervalo especificado.

```
int  Minimum(
   const int  buffer_num,     // número del búfer
   const int  start,          // índice de inicio
   const int  count           // número de elementos
   ) const
```

Parámetros

buffer_num

[in] Número de búfer donde buscar el valor.

start

[in] Índice de inicio de la búsqueda.

count

[in] Número de elementos a buscar.

Valor devuelto

Índice del elemento mínimo del búfer en el intervalo especificado.

© 2000-2025, MetaQuotes Ltd.

MinValue

Devuelve el valor y el índice del elemento mínimo del búfer en el intervalo especificado.

```
double  MinValue(
   const int   buffer_num,     // número de búfer
   const int   start,          // índice de inicio
   const int   count,          // número de elementos
   int&        index           // referencia
   ) const
```

Parámetros

buffer_num

[in] Número de búfer donde buscar el valor.

start

[in] Índice de inicio.

count

[in] Número de elementos.

index

[out] Referencia a la variable de tipo int del índice del elemento mínimo.

Valor devuelto

El valor del elemento mínimo del búfer en el intervalo especificado.

Nota

El índice del elemento mínimo del búfer se almacena en la variable index, pasada por referencia.

© 2000-2025, MetaQuotes Ltd.

Maximum

Devuelve el índice del elemento máximo del búfer en el intervalo especificado.

```
int  Maximum(
   const int  buffer_num,     // número del búfer
   const int  start,          // índice de inicio
   const int  count           // número de elementos
   ) const
```

Parámetros

buffer_num

[in] Número de búfer donde buscar el valor.

start

[in] Índice de inicio de la búsqueda.

count

[in] Número de elementos a buscar.

Valor devuelto

Índice del elemento máximo del búfer en el intervalo especificado.

© 2000-2025, MetaQuotes Ltd.

MaxValue

Devuelve el valor y el índice del elemento máximo del búfer en el intervalo especificado.

```
double  MaxValue(
   const int   buffer_num,     // número de búfer
   const int   start,          // índice de inicio
   const int   count,          // número de elementos
   int&        index           // referencia
   ) const
```

Parámetros

buffer_num

[in] Número de búfer donde buscar el valor.

start

[in] Índice de inicio.

count

[in] Número de elementos.

index

[out] Referencia a la variable de tipo int del índice del elemento máximo.

Valor devuelto

El valor del elemento máximo del búfer en el rango especificado.

Nota

El índice del elemento máximo del búfer se almacena en la variable index, pasada por referencia.

© 2000-2025, MetaQuotes Ltd.

MethodDescription

La función devuelve el valor de la enumeración ENUM_MA_METHOD en formato string.

```
string  MethodDescription(
   const int  val      // valor
   ) const
```

Parámetros

val

[in] Valor de la enumeración ENUM_MA_METHOD.

Valor devuelto

El valor de la enumeración ENUM_MA_METHOD en formato string.

© 2000-2025, MetaQuotes Ltd.

PriceDescription

El método devuelve el valor de la enumeración ENUM_APPLIED_PRICE en formato string.

```
string  PriceDescription(
   const int  val      // valor
   ) const
```

Parámetros

val

[in] Valor de la enumeración ENUM_APPLIED_PRICE.

Valor devuelto

El valor de la enumeración ENUM_APPLIED_PRICE en formato string.

© 2000-2025, MetaQuotes Ltd.

VolumeDescription

El método devuelve el valor de la enumeración ENUM_APPLIED_VOLUME en formato string.

```
string  VolumeDescription(
   const int  val      // valor
   ) const
```

Parámetros

val

[in] Valor de la enumeración ENUM_APPLIED_VOLUME.

Valor devuelto

El valor de la enumeración ENUM_APPLIED_VOLUME en formato string.

© 2000-2025, MetaQuotes Ltd.

AddToChart

Añade el indicador al gráfico.

```
bool  AddToChart(
   const long  chart,      // ID del gráfico
   const int  subwin       // subventana del gráfico
   )
```

Parámetros

chart

[in] ID del gráfico.

subwin

[in] Subventana del gráfico.

Valor devuelto

true - si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

DeleteFromChart

Borra el indicador del gráfico.

```
bool  DeleteFromChart(
   const long  chart,      // ID del gráfico
   const int  subwin       // subventana del gráfico
   )
```

Parámetros

chart

[in] ID del gráfico.

subwin

[in] Subventana del gráfico.

Valor devuelto

true - si se ejecuta correctamente, false en caso de error.

© 2000-2025, MetaQuotes Ltd.

CIndicators

La clase CIndicators es una colección de instancias de períodos de tiempo e indicadores técnicos.

Descripción

La clase CIndicators proporciona métodos para crear indicadores técnicos, almacenarlos y gestionarlos (sincronización de datos, manejadores y gestión de memoria).

Declaración

```
class CIndicators: public CArrayObj
```

Título

```
#include <Indicators\Indicators.mqh>
```

Métodos de la clase

Métodos de creación	
Create	Crea el indicador técnico
Métodos de actualización de datos	
Refresh	Actualiza los datos de todos los indicadores técnicos de la colección

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados.

```
CIndicator*  Create(
   const string           symbol,     // Nombre del símbolo
   const ENUM_TIMEFRAMES  period,     // Periodo
   const ENUM_INDICATOR   type,       // Tipo de indicador
   const int              count,      // Número de parámetros
   const MqlParam&        params      // Referencia al array de parámetros
   )
```

Parámetros

symbol

[in] Nombre del símbolo.

period

[in] Periodo (enumeración ENUM_TIMEFRAMES).

type

[in] Tipo del indicador (ENUM_INDICATOR).

count

[in] Número de parámetros del indicador.

params

[in] Referencia al array de parámetros del indicador.

Valor devuelto

Si se ejecuta correctamente devuelve la referencia al indicador creado, y NULL si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza los datos de todos los indicadores técnicos de la colección.

```
int  Refresh()
```

Valor devuelto

Devuelve las banderas del periodo de tiempo actualizado (banderas de visibilidad del objeto).

© 2000-2025, MetaQuotes Ltd.

Clases de series temporales

Los siguientes capítulos contienen detalles técnicos de las clases de series temporales de la Librería Estándar MQL5, así como la descripción de sus componentes clave.

Class	Descripción
CiSpread	Proporciona acceso a los datos históricos del spread
CiTime	Proporciona acceso al historial de las horas de apertura de las barras
CiTickVolume	Proporciona acceso al historial de los ticks del volumen de las barras
CiRealVolume	Proporciona acceso al historial de los volúmenes reales de las barras
CiOpen	Proporciona acceso al historial de los precios de apertura de las barras
CiHigh	Proporciona acceso al historial de los precios máximos de las barras
CiLow	Proporciona acceso al historial de los precios mínimos de las barras
CiClose	Proporciona acceso al historial de los precios de cierre de las barras

© 2000-2025, MetaQuotes Ltd.

CiSpread

La clase CiSpread está diseñada para acceder a los spreads de las barras del historial.

Descripción

La clase CiSpread proporciona acceso a los spreads de las barras del historial.

Declaración

```
class CiSpread: public CSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CiSpread

Métodos de la clase

Métodos de creación	
Create	Crea una serie temporal
BufferResize	Establece el tamaño del búfer
Métodos de acceso a los datos	
GetData	Obtiene los datos
Métodos de actualización de datos	
Refresh	Actualiza los datos

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

Crea una serie temporal con los parámetros especificados para acceder al historial de los spreads.

```
bool  Create(
   string           symbol,     // símbolo
   ENUM_TIMEFRAMES  period      // periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Período temporal (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si la serie temporal no se puede crear.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Establece el nuevo tamaño de la serie.

```
virtual void  BufferResize(
   int  size      // nuevo tamaño
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene el elemento de la serie temporal por su índice.

```
int  GetData(
   int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la serie temporal, o 0.

GetData

Obtiene el elemento de la serie temporal a partir de la posición inicial y del número de elementos.

```
int  GetData(
   int   start_pos,     // posición inicial
   int   count,         // número de elementos a obtener
   int&  buffer         // array destino
   ) const
```

Parámetros

start_pos

[in] Posición inicial de la serie temporal.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array de datos destino.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal a partir de la hora inicial y del número de elementos.

```
int  GetData(
   datetime  start_time,     // hora de inicio
   int       count,          // número de elementos
   int&      buffer          // array destino
   ) const
```

Parámetros

start_time

© 2000-2025, MetaQuotes Ltd.

[in] Hora de inicio.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array destino de datos.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal por horas de inicio y fin.

```
int  GetData(
   datetime  start_time,     // hora de inicio
   datetime  stop_time,      // hora de fin
   int&      buffer          // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

stop_time

[in] Hora de parada.

buffer

[in] Referencia al array destino de datos

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza los datos de la serie temporal.

```
virtual void  Refresh(
   int  flags      // banderas
   )
```

Parámetros

flags

[in] Banderas del periodo de tiempo.

© 2000-2025, MetaQuotes Ltd.

CiTime

La clase CiTime está diseñada para acceder a las horas de apertura de las barras del historial.

Descripción

La clase CiTime proporciona acceso a las horas de apertura de las barras del historial.

Declaración

```
class CiTime: public CSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CiTime

Métodos de la clase

Métodos de creación	
Create	Crea una serie temporal
BufferResize	Establece el tamaño del búfer
Métodos de acceso a los datos	
GetData	Obtiene los datos
Métodos de actualización de datos	
Refresh	Actualiza los datos

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

Crea una serie temporal con los parámetros especificados para acceder a las horas de apertura de las barras del historial.

```
bool  Create(
   string           symbol,      // símbolo
   ENUM_TIMEFRAMES  period       // periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si la serie temporal no se puede crear.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Establece el nuevo tamaño de la serie.

```
virtual void  BufferResize(
   int  size      // nuevo tamaño
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene el elemento de la serie temporal por su índice.

```
datetime  GetData(
   int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la serie temporal, o 0.

GetData

Obtiene el elemento de la serie temporal a partir de la posición inicial y del número de elementos.

```
int  GetData(
   int   start_pos,     // posición inicial
   int   count,         // número de elementos a obtener
   long& buffer         // array destino
   ) const
```

Parámetros

start_pos

[in] Posición inicial de la serie temporal.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array de datos destino.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal a partir de la hora inicial y del número de elementos.

```
int  GetData(
   datetime  start_time,     // hora de inicio
   int       count,          // número de elementos
   long&     buffer          // array destino
   ) const
```

Parámetros

start_time

© 2000-2025, MetaQuotes Ltd.

[in] Hora de inicio.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array destino de datos.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal por horas de inicio y fin.

```
int  GetData(
   datetime  start_time,     // hora de inicio
   datetime  stop_time,      // hora de fin
   long&     buffer          // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

stop_time

[in] Hora de parada.

buffer

[in] Referencia al array destino de datos

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza los datos de la serie temporal.

```
virtual void  Refresh(
   int  flags      // banderas
   )
```

Parámetros

flags

[in] Banderas del periodo de tiempo.

© 2000-2025, MetaQuotes Ltd.

CiTickVolume

La clase CiTickVolume está diseñada para acceder a los volúmenes de los ticks de las barras del historial.

Descripción

La clase CiTickVolume proporciona acceso a los volúmenes de los ticks de las barras del historial.

Declaración

```
class CiTickVolume: public CSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CiTickVolume

Métodos de la clase

Métodos de creación	
Create	Crea una serie temporal
BufferResize	Establece el tamaño del búfer
Métodos de acceso a los datos	
GetData	Obtiene los datos
Métodos de actualización de datos	
Refresh	Actualiza los datos

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

Crea una serie temporal con los parámetros especificados para acceder a los volúmenes de los ticks de las barras del historial.

```
bool  Create(
   string           symbol,      // símbolo
   ENUM_TIMEFRAMES  period       // periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si la serie temporal no se puede crear.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Establece el nuevo tamaño de la serie.

```
virtual void  BufferResize(
   int  size      // nuevo tamaño
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene el elemento de la serie temporal por su índice.

```
long  GetData(
   int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la serie temporal, o 0.

GetData

Obtiene el elemento de la serie temporal a partir de la posición inicial y del número de elementos.

```
int  GetData(
   int   start_pos,     // posición inicial
   int   count,         // número de elementos a obtener
   long& buffer         // array destino
   ) const
```

Parámetros

start_pos

[in] Posición inicial de la serie temporal.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array de datos destino.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal a partir de la hora inicial y del número de elementos.

```
int  GetData(
   datetime  start_time,     // hora de inicio
   int       count,          // número de elementos
   long&     buffer          // array destino
   ) const
```

Parámetros

start_time

© 2000-2025, MetaQuotes Ltd.

[in] Hora de inicio.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array destino de datos.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal por horas de inicio y fin.

```
int  GetData(
   datetime  start_time,     // hora de inicio
   datetime  stop_time,      // hora de fin
   long&     buffer          // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

stop_time

[in] Hora de parada.

buffer

[in] Referencia al array destino de datos

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza los datos de la serie temporal.

```
virtual void  Refresh(
   int  flags      // banderas
   )
```

Parámetros

flags

[in] Banderas del periodo de tiempo.

© 2000-2025, MetaQuotes Ltd.

CiRealVolume

La clase CiRealVolume está diseñada para acceder a los volúmenes reales de las barras del historial.

Descripción

La clase CiRealVolume proporciona acceso a los volúmenes reales de las barras del historial.

Declaración

```
class CiRealVolume: public CSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CiRealVolume

Métodos de la clase

Métodos de creación	
Create	Crea una serie temporal
BufferResize	Establece el tamaño del búfer
Métodos de acceso a los datos	
GetData	Obtiene los datos
Métodos de actualización de datos	
Refresh	Actualiza los datos

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

Crea una serie temporal con los parámetros especificados para acceder a los volúmenes reales de las barras del historial.

```
bool  Create(
   string           symbol,      // símbolo
   ENUM_TIMEFRAMES  period       // periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si la serie temporal no se puede crear.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Establece el nuevo tamaño de la serie.

```
virtual void  BufferResize(
   int  size      // nuevo tamaño
   )
```

Parámetros

size

[in] Nuevo tamaño del búfer.

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene el elemento de la serie temporal por su índice.

```
datetime  GetData(
   int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la serie temporal, o 0.

GetData

Obtiene el elemento de la serie temporal a partir de la posición inicial y del número de elementos.

```
int  GetData(
   int   start_pos,     // posición inicial
   int   count,         // número de elementos a obtener
   long& buffer         // array destino
   ) const
```

Parámetros

start_pos

[in] Posición inicial de la serie temporal.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array de datos destino.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal a partir de la hora inicial y del número de elementos.

```
int  GetData(
   datetime  start_time,     // hora de inicio
   int       count,          // número de elementos
   long&     buffer          // array destino
   ) const
```

Parámetros

start_time

© 2000-2025, MetaQuotes Ltd.

[in] Hora de inicio.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array destino de datos.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal por horas de inicio y fin.

```
int  GetData(
   datetime  start_time,     // hora de inicio
   datetime  stop_time,      // hora de fin
   long&     buffer          // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

stop_time

[in] Hora de parada.

buffer

[in] Referencia al array destino de datos

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza los datos de la serie temporal.

```
virtual void  Refresh(
   int  flags      // banderas
   )
```

Parámetros

flags

[in] Banderas del periodo de tiempo.

© 2000-2025, MetaQuotes Ltd.

CiOpen

La clase CiOpen está diseñada para acceder a los precios de apertura de las barras del historial.

Descripción

La clase CiOpen proporciona acceso a los precios de apertura de las barras del historial.

Declaración

```
class CiOpen: public CPriceSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiOpen

Métodos de la clase

Métodos de creación	
Create	Crea una serie temporal
Métodos de acceso a los datos	
GetData	Obtiene los datos

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

Crea una serie temporal con los parámetros especificados para acceder a los precios de apertura de las barras del historial.

```
bool  Create(
   string           symbol,      // símbolo
   ENUM_TIMEFRAMES  period       // periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si la serie temporal no se puede crear.

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene el elemento de la serie temporal a partir de la posición inicial y del número de elementos.

```
int  GetData(
   int    start_pos,      // posición inicial
   int    count,          // número de elementos a obtener
   double& buffer         // array destino
   ) const
```

Parámetros

start_pos

[in] Posición inicial de la serie temporal.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array de datos destino.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal a partir de la hora inicial y del número de elementos.

```
int  GetData(
   datetime  start_time,      // hora de inicio
   int       count,           // número de elementos
   double&     buffer         // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array destino de datos.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal por horas de inicio y fin.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // hora de inicio
   datetime  stop_time,       // hora de fin
   double&   buffer           // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

stop_time

[in] Hora de parada.

buffer

[in] Referencia al array destino de datos

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

© 2000-2025, MetaQuotes Ltd.

CiHigh

La clase CiHigh está diseñada para acceder a los precios máximos de las barras del historial.

Descripción

La clase CiHigh proporciona acceso a los precios máximos de las barras del historial.

Declaración

```
class CiHigh: public CPriceSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiHigh

Métodos de la clase

Métodos de creación	
Create	Crea una serie temporal
Métodos de acceso a los datos	
GetData	Obtiene los datos

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

Crea una serie temporal con los parámetros especificados para acceder a los precios máximos de las barras del historial.

```
bool  Create(
   string           symbol,      // símbolo
   ENUM_TIMEFRAMES  period       // periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si la serie temporal no se puede crear.

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene el elemento de la serie temporal a partir de la posición inicial y del número de elementos.

```
int  GetData(
   int    start_pos,      // posición inicial
   int    count,          // número de elementos a obtener
   double& buffer         // array destino
   ) const
```

Parámetros

start_pos

[in] Posición inicial de la serie temporal.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array de datos destino.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal a partir de la hora inicial y del número de elementos.

```
int  GetData(
   datetime  start_time,      // hora de inicio
   int       count,           // número de elementos
   double&     buffer         // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array destino de datos.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal por horas de inicio y fin.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // hora de inicio
   datetime  stop_time,       // hora de fin
   double&   buffer           // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

stop_time

[in] Hora de parada.

buffer

[in] Referencia al array destino de datos

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

© 2000-2025, MetaQuotes Ltd.

CiLow

La clase CiLow está diseñada para acceder a los precios mínimos de las barras del historial.

Descripción

La clase CiLow proporciona acceso a los precios mínimos de las barras del historial.

Declaración

```
class CiLow: public CPriceSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiLow

Métodos de la clase

Métodos de creación	
Create	Crea una serie temporal
Métodos de acceso a los datos	
GetData	Obtiene los datos

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

Crea una serie temporal con los parámetros especificados para acceder a los precios mínimos de las barras del historial.

```
bool  Create(
   string           symbol,      // símbolo
   ENUM_TIMEFRAMES  period       // periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si la serie temporal no se puede crear.

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene el elemento de la serie temporal a partir de la posición inicial y del número de elementos.

```
int  GetData(
   int    start_pos,      // posición inicial
   int    count,          // número de elementos a obtener
   double& buffer         // array destino
   ) const
```

Parámetros

start_pos

[in] Posición inicial de la serie temporal.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array de datos destino.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal a partir de la hora inicial y del número de elementos.

```
int  GetData(
   datetime  start_time,      // hora de inicio
   int       count,           // número de elementos
   double&     buffer         // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array destino de datos.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal por horas de inicio y fin.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // hora de inicio
   datetime  stop_time,       // hora de fin
   double&   buffer           // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

stop_time

[in] Hora de parada.

buffer

[in] Referencia al array destino de datos

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

© 2000-2025, MetaQuotes Ltd.

CiClose

La clase CiClose está diseñada para acceder a los precios de cierre de las barras del historial.

Descripción

La clase CiClose proporciona acceso a los precios de cierre de las barras del historial.

Declaración

```
class CiClose: public CPriceSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiClose

Métodos de la clase

Métodos de creación	
Create	Crea una serie temporal
Métodos de acceso a los datos	
GetData	Obtiene los datos

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

Crea una serie temporal con los parámetros especificados para acceder a los precios de cierre de las barras del historial.

```
bool  Create(
   string           symbol,      // símbolo
   ENUM_TIMEFRAMES  period       // periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si la serie temporal no se puede crear.

© 2000-2025, MetaQuotes Ltd.

GetData

Obtiene el elemento de la serie temporal a partir de la posición inicial y del número de elementos.

```
int  GetData(
   int    start_pos,      // posición inicial
   int    count,          // número de elementos a obtener
   double& buffer         // array destino
   ) const
```

Parámetros

start_pos

[in] Posición inicial de la serie temporal.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array de datos destino.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal a partir de la hora inicial y del número de elementos.

```
int  GetData(
   datetime  start_time,      // hora de inicio
   int       count,           // número de elementos
   double&     buffer         // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

count

[in] Número de elementos requeridos.

buffer

[in] Referencia al array destino de datos.

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

GetData

Obtiene el elemento de la serie temporal por horas de inicio y fin.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // hora de inicio
   datetime  stop_time,       // hora de fin
   double&   buffer           // array destino
   ) const
```

Parámetros

start_time

[in] Hora de inicio.

stop_time

[in] Hora de parada.

buffer

[in] Referencia al array destino de datos

Valor devuelto

>=0 en caso de éxito, -1 en caso de error.

© 2000-2025, MetaQuotes Ltd.

Clases de los indicadores de tendencias

Los siguientes capítulos contienen detalles técnicos de las clases indicadoras de tendencias de la Librería Estándar MQL5, así como la descripción de sus componentes clave.

Clase/grupo	Descripción
CiADX	Average Directional Index (Índice Direccional Medio)
CiADXWilder	Average Directional Index (Índice Direccional Medio) de Welles Wilder
CiBands	Bandas de Bollinger®
CiEnvelopes	Envelopes
CiIchimoku	Ichimoku Kinko Hyo
CiMA	Media Móvil
CiSAR	Stop Parabólico y Sistema Inverso
CiStdDev	Desviación Estándar
CiDEMA	Media Móvil Exponencial Doble
CiTEMA	Media Móvil Exponencial Triple
CiFrAMA	Media Móvil Adaptativa Fractal
CiAMA	Media Móvil Adaptativa
CiVIDyA	Índice Variable de Media Dinámica

© 2000-2025, MetaQuotes Ltd.

CiADX

La clase CiADX está diseñada para utilizar el indicador Average Directional Index (Índice Direccional Medio).

Descripción

La clase CiADX proporciona métodos para la creación y el acceso a los datos del indicador Average Directional Index (Índice Direccional Medio).

Declaración

```
class CiADX: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADX

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento del búfer de la línea principal
Plus	Devuelve el elemento del búfer de la línea +DI
Minus	Devuelve el elemento del búfer de la línea -DI
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,        // Símbolo
   ENUM_TIMEFRAMES  period,        // Periodo
   int              ma_period      // Periodo de promediación
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Plus

Devuelve el elemento del búfer de la línea +DI por el índice especificado.

```
double   Plus(
   int   index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea +DI del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Minus

Devuelve el elemento del búfer de la línea -DI por el índice especificado.

```
double  Minus(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea -DI del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_ADX para CiADX).

© 2000-2025, MetaQuotes Ltd.

CiADXWilder

La clase CiADXWilder está diseñada para utilizar el indicador técnico Average Directional Index (Índice Direccional Medio) de Welles Wilder.

Descripción

La clase CiADXWilder proporciona métodos para la creación y el acceso a los datos del indicador Average Directional Index (Índice Direccional Medio) de Welles Wilder.

Declaración

```
class CiADXWilder: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADXWilder

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento del búfer de la línea principal
Plus	Devuelve el elemento del búfer de la línea +DI
Minus	Devuelve el elemento del búfer de la línea -DI
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,        // Símbolo
   ENUM_TIMEFRAMES  period,        // Periodo
   int              ma_period      // Periodo de promediación
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Plus

Devuelve el elemento del búfer de la línea +DI por el índice especificado.

```
double  Plus(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea +DI del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Minus

Devuelve el elemento del búfer de la línea -DI por el índice especificado.

```
double  Minus(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea -DI del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_ADXW para CiADXWilder).

© 2000-2025, MetaQuotes Ltd.

CiBands

La clase CiBands está diseñada para utilizar el indicador técnico Bollinger Bands®.

Descripción

La clase CiBands proporciona métodos de creación y acceso a los datos del indicador Bollinger Bands®.

Declaración

```
class CiBands: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBands

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
MaShift	Devuelve el desplazamiento horizontal
Deviation	Devuelve la desviación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Base	Devuelve el elemento del búfer de la línea base
Upper	Devuelve el elemento del búfer de la línea superior
Lower	Devuelve el elemento del búfer de la línea inferior
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaShift

Devuelve el desplazamiento horizontal del indicador.

```
int  MaShift() const
```

Valor devuelto

Devuelve el valor del desplazamiento horizontal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Deviation

Devuelve la desviación.

```
double  Deviation() const
```

Valor devuelto

Devuelve la desviación, definida en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,         // Símbolo
   ENUM_TIMEFRAMES  period,         // Periodo
   int              ma_period,      // Periodo de promediación
   int              ma_shift,       // Desplazamiento
   double           deviation,      // Desviación
   int              applied         // precio aplicado, o manejador
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

ma_shift

[in] Desplazamiento horizontal del indicador.

deviation

[in] Desviación.

applied

[in] Tipo de volumen a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Base

Devuelve el elemento del búfer de la línea base por el índice especificado.

```
double  Base(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea Base del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Upper

Devuelve el elemento del búfer de la línea superior por el índice especificado.

```
double  Upper(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea superior del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Lower

Devuelve el elemento del búfer de la línea inferior por el índice especificado.

```
double  Lower(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea inferior del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_BANDS para CiBands).

© 2000-2025, MetaQuotes Ltd.

CiEnvelopes

La clase CiEnvelopes está diseñada para utilizar el indicador técnico Envelopes.

Descripción

La clase CiEnvelopes proporciona métodos para la creación y acceso a los datos del indicador Envelopes.

Declaración

```
class CiEnvelopes: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiEnvelopes

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
MaShift	Devuelve el desplazamiento horizontal
MaMethod	Devuelve el método de promediación
Deviation	Devuelve la desviación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Upper	Devuelve el elemento del búfer de la línea superior
Lower	Devuelve el elemento del búfer de la línea inferior
Entrada/salida	

© 2000-2025, MetaQuotes Ltd.

Atributos	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaShift

Devuelve el desplazamiento horizontal del indicador.

```
int  MaShift() const
```

Valor devuelto

Devuelve el valor del desplazamiento horizontal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Devuelve el método de promediación.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor devuelto

Devuelve el método de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Deviation

Devuelve el valor de la desviación.

```
double  Deviation() const
```

Valor devuelto

Devuelve el valor de la desviación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,        // Símbolo
   ENUM_TIMEFRAMES  period,        // Periodo
   int              ma_period,     // Periodo de promediación
   int              ma_shift,      // Desplazamiento horizontal
   ENUM_MA_METHOD   ma_method,     // Método de promediación
   int              applied,       // Tipo de precio o manejador a aplicar
   double           deviation      // Desviación
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

ma_shift

[in] Desplazamiento horizontal.

ma_method

[in] Método de promediación (enumeración ENUM_MA_METHOD).

applied

[in] Tipo de precio o manejador a aplicar.

deviation

[in] Desviación.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Upper

Devuelve el elemento del búfer de la línea superior por el índice especificado.

```
double  Upper(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea superior del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Lower

Devuelve el elemento del búfer de la línea inferior por el índice especificado.

```
double  Lower(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea inferior del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_ENVELOPES es el de CiEnvelopes).

© 2000-2025, MetaQuotes Ltd.

CiIchimoku

La clase CiIchimoku está diseñada para utilizar el indicador técnico Ichimoku Kinko Hyo.

Descripción

La clase CiIchimoku proporciona métodos para la creación, configuración y acceso a los datos del indicador Ichimoku Kinko Hyo.

Declaración

```
class CiIchimoku: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiIchimoku

Métodos de la clase

Atributos	
TenkanSenPeriod	Devuelve el periodo TenkanSen
KijunSenPeriod	Devuelve el periodo KijunSen
SenkouSpanBPeriod	Devuelve el periodo SenkouSpanB
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
TenkanSen	Devuelve el elemento del búfer de la línea TenkanSen
KijunSen	Devuelve el elemento del búfer de la línea KijunSen
SenkouSpanA	Devuelve el elemento del búfer de la línea SenkouSpanA
SenkouSpanB	Devuelve el elemento del búfer de la línea SenkouSpanB

© 2000-2025, MetaQuotes Ltd.

Atributos	
ChinkouSpan	Devuelve el elemento del búfer de la línea ChikouSpan
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

TenkanSenPeriod

Devuelve el periodo TenkanSen.

```
int  TenkanSenPeriod() const
```

Valor devuelto

Devuelve el periodo TenkanSen, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

KijunSenPeriod

Devuelve el periodo KijunSen.

```
int  KijunSenPeriod() const
```

Valor devuelto

Devuelve el periodo KijunSen, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

SenkouSpanBPeriod

Devuelve el periodo SenkouSpanB.

```
int  SenkouSpanBPeriod() const
```

Valor devuelto

Devuelve el periodo SenkouSpanB, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,            // Símbolo
   ENUM_TIMEFRAMES  period,            // Periodo
   int              tenkan_sen,        // Periodo de TenkanSen
   int              kijun_sen,         // Periodo de KijunSen
   int              senkou_span_b      // Periodo de SenkouSpanB
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

tenkan_sen

[in] Periodo de TenkanSen.

kijun_sen

[in] Periodo de KijunSen.

senkou_span_b

[in] Periodo de SenkouSpanB.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

TenkanSen

Devuelve el elemento del búfer de la línea TenkanSen por el índice especificado.

```
double  TenkanSen(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea TenkanSen del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

KijunSen

Devuelve el elemento del búfer de la línea KijunSen por el índice especificado.

```
double  KijunSen(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea KijunSen del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

SenkouSpanA

Devuelve el elemento del búfer de la línea SenkouSpanA por el índice especificado.

```
double  SenkouSpanA(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea SenkouSpanA del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

SenkouSpanB

Devuelve el elemento del búfer de la línea SenkouSpanB por el índice especificado.

```
double  SenkouSpanB(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea SenkouSpanB del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

ChinkouSpan

Devuelve el elemento del búfer de la línea ChinkouSpan por el índice especificado.

```
double  ChinkouSpan(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea ChinkouSpan del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_ICHIMOKU para CiIchimoku).

© 2000-2025, MetaQuotes Ltd.

CiMA

La clase CiMA está diseñada para utilizar el indicador técnico Media Móvil.

Descripción

La clase CiMA proporciona métodos para la creación, configuración y acceso a los datos del indicador Media Móvil.

Declaración

```
class CiMA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMA

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
MaShift	Devuelve el desplazamiento horizontal
MaMethod	Devuelve el método de promediación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaShift

Devuelve el desplazamiento horizontal del indicador.

```
int  MaShift() const
```

Valor devuelto

Devuelve el valor del desplazamiento horizontal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Devuelve el método de promediación.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor devuelto

Devuelve el método de promediación (valor de la enumeración ENUM_MA_METHOD), definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           string,        // Símbolo
   ENUM_TIMEFRAMES  period,        // Periodo
   int              ma_period,     // Periodo de promediación
   int              ma_shift,      // Desplazamiento horizontal
   ENUM_MA_METHOD   ma_method,     // Método de promediación
   int              applied        // Tipo de precio del manejador a aplicar
   )
```

Parámetros

string

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

ma_shift

[in] Desplazamiento horizontal.

ma_method

[in] Método de promediación (enumeración ENUM_MA_METHOD).

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_MA es el de CiMA).

© 2000-2025, MetaQuotes Ltd.

CiSAR

La clase CiSAR está diseñada para utilizar el indicador técnico Stop Parabólico y Sistema Inverso.

Descripción

La clase CiSAR proporciona métodos para la creación, configuración y acceso a los datos del indicador Stop Parabólico y Sistema Inverso.

Declaración

```
class CiSAR: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiSAR

Métodos de la clase

Atributos	
SarStep	Devuelve el paso del aumento de velocidad
Maximum	Devuelve el coeficiente del precio siguiente
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

SarStep

Devuelve el paso del aumento de velocidad (coeficiente de aceleración).

```
double  SarStep() const
```

Valor devuelto

El paso del aumento de velocidad (coeficiente de aceleración), definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Maximum

Devuelve el coeficiente del precio siguiente.

```
double  Maximum() const
```

Valor devuelto

El coeficiente del precio siguiente, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,      // Símbolo
   ENUM_TIMEFRAMES  period,      // Periodo
   double           step,        // Paso
   double           maximum      // Coeficiente
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

step

[in] Paso del incremento de velocidad.

maximum

[in] Coeficiente del precio siguiente.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_SAR para CiSAR).

© 2000-2025, MetaQuotes Ltd.

CiStdDev

La clase CiStdDev está diseñada para utilizar el indicador técnico Desviación Estándar.

Descripción

La clase CiStdDev proporciona métodos para la creación, configuración y acceso a los datos del indicador Desviación Estándar.

Declaración

```
class CiStdDev: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiStdDev

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
MaShift	Devuelve el desplazamiento horizontal
MaMethod	Devuelve el método de promediación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaShift

Devuelve el desplazamiento horizontal del indicador.

```
int  MaShift() const
```

Valor devuelto

Devuelve el valor del desplazamiento horizontal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Devuelve el método de promediación.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor devuelto

Devuelve el método de promediación (valor de la enumeración ENUM_MA_METHOD), definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string            symbol,         // Símbolo
   ENUM_TIMEFRAMES   period,         // Periodo
   int               ma_period,      // Periodo de promediación
   int               ma_shift,       // Desplazamiento horizontal
   ENUM_MA_METHOD    ma_method,      // Método de promediación
   int               applied         // Tipo de precio o manejador a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

ma_shift

[in] Desplazamiento horizontal.

ma_method

[in] Método de promediación (enumeración ENUM_MA_METHOD).

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_STDDEV es el de CiStdDev).

© 2000-2025, MetaQuotes Ltd.

CiDEMA

La clase CiDEMA está diseñada para utilizar el indicador técnico Media Móvil Exponencial Doble.

Descripción

La clase CiDEMA proporciona métodos para la creación, configuración y acceso a los datos del indicador Media Móvil Exponencial Doble.

Declaración

```
class CiDEMA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiDEMA

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
IndShift	Devuelve el desplazamiento horizontal
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

IndShift

Devuelve el desplazamiento horizontal del indicador.

```
int  IndShift() const
```

Valor devuelto

Devuelve el valor del desplazamiento horizontal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           string,        // Símbolo
   ENUM_TIMEFRAMES  period,        // Periodo
   int              ma_period,     // Periodo de promediación
   int              ind_shift,     // Desplazamiento
   int              applied        // Tipo de precio del manejador a aplicar
   )
```

Parámetros

string

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

ind_shift

[in] Desplazamiento horizontal.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_DEMA para CiDEMA).

© 2000-2025, MetaQuotes Ltd.

CiTEMA

La clase CiTEMA está diseñada para utilizar el indicador técnico Media Móvil Exponencial Triple.

Descripción

La clase CiTEMA proporciona métodos para la creación, configuración y acceso a los datos del indicador Media Móvil Exponencial Triple.

Declaración

```
class CiTEMA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTEMA

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
IndShift	Devuelve el desplazamiento horizontal
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

IndShift

Devuelve el desplazamiento horizontal del indicador.

```
int  IndShift() const
```

Valor devuelto

Devuelve el valor del desplazamiento horizontal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,         // Símbolo
   ENUM_TIMEFRAMES  period,         // Periodo
   int              ma_period,      // Periodo de promediación
   int              ma_shift,       // Offset
   int              applied         // Tipo de precio del manejador a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

ma_shift

[in] Desplazamiento horizontal.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_TEMA para CiTEMA).

© 2000-2025, MetaQuotes Ltd.

CiFrAMA

La clase CiFrAMA está diseñada para utilizar el indicador técnico Media Móvil Adaptativa Fractal.

Descripción

La clase CiFrAMA proporciona métodos para la creación, configuración y acceso a los datos del indicador Media Móvil Adaptativa Fractal.

Declaración

```
class CiFrAMA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiFrAMA

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
IndShift	Devuelve el desplazamiento horizontal
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

IndShift

Devuelve el desplazamiento horizontal del indicador.

```
int  IndShift() const
```

Valor devuelto

Devuelve el valor del desplazamiento horizontal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,         // Símbolo
   ENUM_TIMEFRAMES  period,         // Periodo
   int              ma_period,      // Periodo de promediación
   int              ma_shift,       // Offset
   int              applied         // Tipo de precio del manejador a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

ma_shift

[in] Desplazamiento horizontal.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_FRAMA para CiFrAMA).

© 2000-2025, MetaQuotes Ltd.

CiAMA

La clase CiAMA está diseñada para utilizar el indicador técnico Media Móvil Adaptativa.

Descripción

La clase CiAMA proporciona métodos para la creación, configuración y acceso a los datos del indicador Media Móvil Adaptativa.

Declaración

```
class CiAMA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAMA

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
FastEmaPeriod	Devuelve el período de promediación de la EMA rápida
SlowEmaPeriod	Devuelve el período de promediación de la EMA lenta
IndShift	Devuelve el desplazamiento horizontal
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

FastEmaPeriod

Devuelve el periodo de promediación de la EMA rápida.

```
int  FastEmaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la EMA rápida, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

SlowEmaPeriod

Devuelve el periodo de promediación de la EMA lenta.

```
int  SlowEmaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la EMA lenta, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

IndShift

Devuelve el desplazamiento horizontal del indicador.

```
int  IndShift() const
```

Valor devuelto

Devuelve el valor del desplazamiento horizontal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           string,              // Símbolo
   ENUM_TIMEFRAMES  period,              // Periodo
   int              ma_period,           // Periodo de promediación
   int              fast_ema_period,     // Periodo de la EMA rápida
   int              slow_ema_period,     // Periodo de la EMA lenta
   int              ind_shift,           // Desplazamiento
   int              applied              // Tipo de precio o manejador a aplicar
   )
```

Parámetros

string

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

fast_ema_period

[in] Período de promediación de la EMA rápida.

slow_ema_period

[in] Período de promediación de la EMA lenta.

ind_shift

[in] Desplazamiento horizontal.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_AMA es el de CiAMA).

© 2000-2025, MetaQuotes Ltd.

CiVIDyA

La clase CiVIDyA está diseñada para utilizar el indicador técnico Índice Variable de Media Dinámica.

Descripción

La clase CiVIDyA proporciona métodos para la creación, configuración y acceso a los datos del indicador Índice Variable de Media Dinámica.

Declaración

```
class CiVIDyA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiVIDyA

Métodos de la clase

Atributos	
CmoPeriod	Devuelve el periodo de Momentum
EmaPeriod	Devuelve el periodo de promediación de la EMA
IndShift	Devuelve el desplazamiento horizontal
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

CmoPeriod

Devuelve el periodo de Momentum.

```
int  CmoPeriod() const
```

Valor devuelto

Devuelve el periodo de Momentum, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

EmaPeriod

Devuelve el periodo de promediación de la EMA.

```
int  EmaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la EMA, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

IndShift

Devuelve el desplazamiento horizontal del indicador.

```
int  IndShift() const
```

Valor devuelto

Devuelve el valor del desplazamiento horizontal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,          // Símbolo
   ENUM_TIMEFRAMES  period,          // Periodo
   int              cmo_period,      // Periodo de Momentum
   int              ema_period,      // Periodo de promediación
   int              ind_shift,       // Desplazamiento
   int              applied          // Tipo de precio o manejador a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

cmo_period

[in] Periodo de Momentum.

ema_period

[in] Periodo de promediación.

ind_shift

[in] Desplazamiento horizontal.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_VIDYA es el de CiVIDyA).

© 2000-2025, MetaQuotes Ltd.

Osciladores

Los siguientes capítulos contienen detalles técnicos de las clases de los osciladores de la Librería Estándar MQL5, así como la descripción de sus componentes clave.

Clase/grupo	Descripción
CiATR	Average True Range (Rango Medio Verdadero)
CiBearsPower	Bears Power (Poder de los Osos)
CiBullsPower	Bulls Power (Poder de los Toros)
CiCCI	Commodity Channel Index (Índice del Canal de Productos)
CiChaikin	Chaikin Oscillator (Oscilador Chaikin)
CiDeMarker	DeMarker
CiForce	Force Index (Índice de Fuerza)
CiMACD	Moving Averages Convergence-Divergence (Convergencia/Divergencia del Promedio Móvil)
CiMomentum	Momentum
CiOsMA	Media Móvil del Oscilador (Histograma MACD)
CiRSI	Índice de Fuerza Relativa
CiRVI	Índice de Vigor Relativo
CiStochastic	Oscilador Estocástico
CiWPR	Rango Porcentual de Williams
CiTriX	Oscilador Media Móvil Exponencial Triple

© 2000-2025, MetaQuotes Ltd.

CiATR

La clase CiATR está diseñada para utilizar el indicador técnico Average True Range (Rango Medio Verdadero).

Descripción

La clase CiATR proporciona métodos para la creación, configuración y acceso a los datos del indicador Average True Range (Rango Medio Verdadero).

Declaración

```
class CiATR: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiATR

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,        // Símbolo
   ENUM_TIMEFRAMES  period,        // Periodo
   int              ma_period      // Periodo de promediación
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_ATR para CiATR).

© 2000-2025, MetaQuotes Ltd.

CiBearsPower

La clase CiBearsPower está diseñada para utilizar el indicador técnico Bears Power (Poder de los Osos).

Descripción

La clase CiBearsPower proporciona métodos para la creación, configuración y acceso a los datos del indicador Bears Power (Poder de los Osos).

Declaración

```
class CiBearsPower: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBearsPower

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,        // Símbolo
   ENUM_TIMEFRAMES  period,        // Periodo
   int              ma_period      // Periodo de promediación
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_BEARS para CiBearsPower).

© 2000-2025, MetaQuotes Ltd.

CiBullsPower

La clase CiBullsPower está diseñada para utilizar el indicador técnico Bulls Power (Poder de los Toros).

Descripción

La clase CiBullsPower proporciona métodos para la creación, configuración y acceso a los datos del indicador Bulls Power (Poder de los Toros).

Declaración

```
class CiBullsPower: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBullsPower

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,        // Símbolo
   ENUM_TIMEFRAMES  period,        // Periodo
   int              ma_period      // Periodo de promediación
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_BULLS para CiBullsPower).

© 2000-2025, MetaQuotes Ltd.

CiCCI

La clase CiCCI está diseñada para utilizar el indicador técnico Commodity Channel Index (Índice del Canal de Productos).

Descripción

La clase CiCCI proporciona métodos para la creación, configuración y acceso a los datos del indicador Commodity Channel Index (Índice del Canal de Productos).

Declaración

```
class CiCCI: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiCCI

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,         // Símbolo
   ENUM_TIMEFRAMES  period,         // Periodo
   int              ma_period,      // Periodo de promediación
   int              applied         // Tipo de precio o manejador a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_CCI para CiCCI).

© 2000-2025, MetaQuotes Ltd.

CiChaikin

La clase CiChaikin está diseñada para utilizar el indicador técnico Oscilador Chaikin.

Descripción

La clase CiChaikin proporciona métodos para la creación, configuración y acceso a los datos del indicador Oscilador Chaikin.

Declaración

```
class CiChaikin: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiChaikin

Métodos de la clase

Atributos	
FastMaPeriod	Devuelve el periodo de promediación de la MA rápida
SlowMaPeriod	Devuelve el periodo de promediación de la MA lenta
MaMethod	Devuelve el método de promediación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

FastMaPeriod

Devuelve el periodo de promediación de la EMA rápida.

```
int  FastMaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la EMA rápida, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

SlowMaPeriod

Devuelve el periodo de promediación de la EMA lenta.

```
int  SlowMaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la EMA lenta, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Devuelve el método de promediación.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor devuelto

Devuelve el método de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de volumen a aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor devuelto

Tipo de volumen a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string               symbol,             // Símbolo
   ENUM_TIMEFRAMES      period,             // Periodo
   int                  fast_ma_period,     // Periodo de la EMA rápida
   int                  slow_ma_period,     // Periodo de la EMA lenta
   ENUM_MA_METHOD       ma_method,          // Método de promediación
   ENUM_APPLIED_VOLUME  applied             // Tipo de volumen a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

fast_ma_period

[in] Periodo de la EMA rápida.

slow_ma_period

[in] Periodo de la EMA lenta.

ma_method

[in] Método de promediación (enumeración ENUM_MA_METHOD).

applied

[in] Tipo de volumen a aplicar (enumeración ENUM_APPLIED_VOLUME).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_CHAIKIN para CiChaikin).

© 2000-2025, MetaQuotes Ltd.

CiDeMarker

La clase CiDeMarker está diseñada para utilizar el indicador técnico DeMarker.

Descripción

La clase CiDeMarker proporciona métodos para la creación, configuración y acceso a los datos del indicador DeMarker.

Declaración

```
class CiDeMarker: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiDeMarker

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,        // Símbolo
   ENUM_TIMEFRAMES  period,        // Periodo
   int              ma_period      // Periodo de promediación
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_DEMARKER para CiDeMarker).

© 2000-2025, MetaQuotes Ltd.

CiForce

La clase CiForce está diseñada para utilizar el indicador técnico Force Index (Índice de Fuerza).

Descripción

La clase CiForce proporciona métodos para la creación, configuración y acceso a los datos del indicador Force Index (Índice de Fuerza).

Declaración

```
class CiForce: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiForce

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
MaMethod	Devuelve el método de promediación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Devuelve el método de promediación.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor devuelto

Devuelve el método de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de volumen a aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor devuelto

Tipo de volumen a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string                symbol,         // Símbolo
   ENUM_TIMEFRAMES       period,         // Periodo
   int                   ma_period,      // Periodo de promediación
   ENUM_MA_METHOD        ma_method,      // Método de promediación
   ENUM_APPLIED_VOLUME   applied         // Tipo de volumen a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

ma_method

[in] Método de promediación (enumeración ENUM_MA_METHOD).

applied

[in] Tipo de volumen a aplicar (enumeración ENUM_APPLIED_VOLUME).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_FORCE para CiForce).

© 2000-2025, MetaQuotes Ltd.

CiMACD

La clase CiMACD está diseñada para utilizar el indicador técnico Convergencia/Divergencia del Promedio Móvil.

Descripción

La clase CiMACD proporciona métodos para la creación, configuración y acceso a los datos del indicador Convergencia/Divergencia del Promedio Móvil.

Declaración

```
class CiMACD: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMACD

Métodos de la clase

Atributos	
FastEmaPeriod	Devuelve el periodo de promediación de la EMA rápida
SlowEmaPeriod	Devuelve el periodo de promediación de la EMA lenta
SignalPeriod	Devuelve el periodo de promediación de la línea de señal
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento del búfer de la línea principal
Signal	Devuelve el elemento del búfer de la línea de señal
Entrada/salida	

© 2000-2025, MetaQuotes Ltd.

Atributos	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

FastEmaPeriod

Devuelve el periodo de promediación de la EMA rápida.

```
int  FastEmaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la EMA rápida, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

SlowEmaPeriod

Devuelve el periodo de promediación de la EMA lenta.

```
int  SlowEmaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la EMA lenta, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

SignalPeriod

Devuelve el periodo de promediación de la línea de señal.

```
int  SignalPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la línea de señal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,              // Símbolo
   ENUM_TIMEFRAMES  period,              // Periodo
   int              fast_ema_period,     // Periodo de la EMA rápida
   int              slow_ema_period,     // Periodo de la EMA lenta
   int              signal_period,       // Periodo de la señal
   int              applied              // Tipo de precio o manejador a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

fast_ema_period

[in] Periodo de la EMA rápida.

slow_ema_period

[in] Periodo de la EMA lenta.

signal_period

[in] Periodo de la línea de señal.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Signal

Devuelve el elemento del búfer de la línea de señal por el índice especificado.

```
double  Signal(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea de señal del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_MACD es el de CiMACD).

© 2000-2025, MetaQuotes Ltd.

CiMomentum

La clase CiMomentum está diseñada para utilizar el indicador técnico Momentum.

Descripción

La clase CiMomentum proporciona métodos para la creación, configuración y acceso a los datos del indicador Momentum.

Declaración

```
class CiMomentum: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMomentum

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,         // Símbolo
   ENUM_TIMEFRAMES  period,         // Periodo
   int              ma_period,      // Periodo de promediación
   int              applied         // Tipo de precio o manejador a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_MOMENTUM es el de CiMomentum).

© 2000-2025, MetaQuotes Ltd.

CiOsMA

La clase CiOsMA está diseñada para utilizar el indicador técnico Media Móvil del Oscilador (Histograma MACD).

Descripción

La clase CiOsMA proporciona métodos para la creación, configuración y acceso a los datos del indicador Media Móvil del Oscilador (Histograma MACD).

Declaración

```
class CiOsMA: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiOsMA

Métodos de la clase

Atributos	
FastEmaPeriod	Devuelve el periodo de promediación de la EMA rápida
SlowEmaPeriod	Devuelve el periodo de promediación de la EMA lenta
SignalPeriod	Devuelve el periodo de promediación de la línea de señal
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

FastEmaPeriod

Devuelve el periodo de promediación de la EMA rápida.

```
int  FastEmaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la EMA rápida, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

SlowEmaPeriod

Devuelve el periodo de promediación de la EMA lenta.

```
int  SlowEmaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la EMA lenta, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

SignalPeriod

Devuelve el periodo de promediación de la línea de señal.

```
int  SignalPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la línea de señal, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,              // Símbolo
   ENUM_TIMEFRAMES  period,              // Periodo
   int              fast_ema_period,     // Periodo de la EMA rápida
   int              slow_ema_period,     // Periodo de la EMA lenta
   int              signal_period,       // Periodo de la línea de la señal
   int              applied              // Tipo de precio o manejador a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

fast_ema_period

[in] Periodo de la EMA rápida.

slow_ema_period

[in] Periodo de la EMA lenta.

signal_period

[in] Periodo de la línea de señal.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_OSMA para CiOsMA).

© 2000-2025, MetaQuotes Ltd.

CiRSI

La clase CiRSI está diseñada para utilizar el indicador técnico Índice de Fuerza Relativa.

Descripción

La clase CiRSI proporciona métodos para la creación, configuración y acceso a los datos del indicador Índice de Fuerza Relativa.

Declaración

```
class CiRSI: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRSI

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,         // Símbolo
   ENUM_TIMEFRAMES  period,         // Periodo
   int              ma_period,      // Periodo de promediación
   int              applied         // Tipo de precio o manejador a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_RSI es el de CiRSI).

© 2000-2025, MetaQuotes Ltd.

CiRVI

La clase CiRVI está diseñada para utilizar el indicador técnico Índice de Vigor Relativo.

Descripción

La clase CiRVI proporciona métodos para la creación, configuración y acceso a los datos del indicador Índice de Vigor Relativo.

Declaración

```
class CiRVI: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRVI

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento del búfer de la línea base
Signal	Devuelve el elemento del búfer de la línea de señal
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,        // Símbolo
   ENUM_TIMEFRAMES  period,        // Periodo
   int              ma_period      // Periodo de promediación
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Signal

Devuelve el elemento del búfer de la línea de señal por el índice especificado.

```
double  Signal(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea de señal del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_RVI para CiRVI).

© 2000-2025, MetaQuotes Ltd.

CiStochastic

La clase CiStochastic está diseñada para utilizar el indicador técnico Oscilador Estocástico.

Descripción

La clase CiStochastic proporciona métodos para la creación, configuración y acceso a los datos del indicador Oscilador Estocástico.

Declaración

```
class CiStochastic: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiStochastic

Métodos de la clase

Atributos	
Kperiod	Devuelve el periodo de promediación de la línea %K
Dperiod	Devuelve el periodo de promediación de la línea %D
Slowing	Devuelve el periodo de desaceleración
MaMethod	Devuelve el método de promediación
PriceField	Tipo de precio (Bajo/Alto o Cierre/Cierre) a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento del búfer de la línea base
Signal	Devuelve el elemento del búfer de la línea de señal
Entrada/salida	

© 2000-2025, MetaQuotes Ltd.

Atributos	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Kperiod

Devuelve el periodo de promediación de la línea %K.

```
int  Kperiod() const
```

Valor devuelto

Devuelve el periodo de promediación de la línea %K, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Dperiod

Devuelve el periodo de promediación de la línea %D.

```
int  Dperiod() const
```

Valor devuelto

Devuelve el periodo de promediación de la línea %D, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Slowing

Devuelve el periodo de desaceleración.

```
int  Slowing() const
```

Valor devuelto

Devuelve el periodo de desaceleración, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Devuelve el método de promediación.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor devuelto

Devuelve el método de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

PriceField

Devuelve el tipo de precio (Bajo/Alto o Cierre/Cierre) a aplicar.

```
ENUM_STO_PRICE  PriceField() const
```

Valor devuelto

El tipo de precio a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,          // Símbolo
   ENUM_TIMEFRAMES  period,          // Periodo
   int              Kperiod,         // Periodo de promediación de %K
   int              Dperiod,         // Periodo de promediación de %D
   int              slowing,         // Periodo de desaceleración
   ENUM_MA_METHOD   ma_method,       // Método de promediación
   ENUM_STO_PRICE   price_field      // Tipo de precio a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Kperiod

[in] Periodo de promediación de la línea %K.

Dperiod

[in] Periodo de promediación de la línea %D.

slowing

[in] Periodo de desaceleración.

ma_method

[in] Método de promediación (enumeración ENUM_MA_METHOD).

price_field

[in] Tipo de precio (Bajo/Alto o Cierre/Cierre) a aplicar (enumeración ENUM_STO_PRICE).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Signal

Devuelve el elemento del búfer de la línea de señal por el índice especificado.

```
double  Signal(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Signal

Devuelve el elemento del búfer de la línea de señal por el índice especificado.

```
double  Signal(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea de señal del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_STOCHASTIC para CiStochastic).

© 2000-2025, MetaQuotes Ltd.

CiTriX

La clase CiTriX está diseñada para utilizar el indicador técnico Oscilador Media Móvil Exponencial Triple.

Descripción

La clase CiTriX proporciona métodos para la creación, configuración y acceso a los datos del indicador Oscilador Media Móvil Exponencial Triple.

Declaración

```
class CiTriX: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTriX

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,         // Símbolo
   ENUM_TIMEFRAMES  period,         // Periodo
   int              ma_period,      // Periodo de promediación
   int              applied         // Tipo de precio o manejador
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

applied

[in] Tipo de precio o manejador a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_TRIX es el de CiTriX).

© 2000-2025, MetaQuotes Ltd.

CiWPR

La clase CiWPR está diseñada para utilizar el indicador técnico Rango Porcentual de Williams.

Descripción

La clase CiWPR proporciona métodos para la creación, configuración y acceso a los datos del indicador Rango Porcentual de Williams.

Declaración

```
class CiWPR: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiWPR

Métodos de la clase

Atributos	
CalcPeriod	Devuelve el periodo de cálculo
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

CalcPeriod

Devuelve el periodo de cálculo.

```
int  CalcPeriod() const
```

Valor devuelto

Devuelve el periodo de cálculo, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,          // Símbolo
   ENUM_TIMEFRAMES  period,          // Periodo
   int              calc_period      // Periodo de cálculo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

calc_period

[in] Periodo de cálculo.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_WPR para CiWPR).

© 2000-2025, MetaQuotes Ltd.

Indicadores de volumen

Los siguientes capítulos contienen detalles técnicos de las clases indicadoras de volumen de la Librería Estándar MQL5, así como la descripción de sus componentes clave.

Clase/grupo	Descripción
CiAD	Acumulación/Distribución
CiMFI	Índice de Flujo de Dinero
CiOBV	Balance de Volúmenes
CiVolumes	Volúmenes

© 2000-2025, MetaQuotes Ltd.

CiAD

La clase CiAD está diseñada para utilizar el indicador técnico Acumulación/Distribución.

Descripción

La clase CiAD proporciona métodos para la creación, configuración y acceso a los datos del indicador Acumulación/Distribución.

Declaración

```
class CiAD: public CIndicator
```

Título

```
#include <Indicators\Volumes.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAD

Métodos de la clase

Atributos	
Applied	Devuelve el tipo de volumen a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de volumen a aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor devuelto

Tipo de volumen a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string                symbol,      // Símbolo
   ENUM_TIMEFRAMES       period,      // Período
   ENUM_APPLIED_VOLUME   applied      // Tipo de volumen a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

applied

[in] Tipo de volumen a aplicar (enumeración ENUM_APPLIED_VOLUME).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_AD es el de CiAD).

© 2000-2025, MetaQuotes Ltd.

CiMFI

La clase CiMFI está diseñada para utilizar el indicador técnico Índice de Flujo de Dinero.

Descripción

La clase CiMFI proporciona métodos para la creación, configuración y acceso a los datos del indicador Índice de Flujo de Dinero.

Declaración

```
class CiMFI: public CIndicator
```

Título

```
#include <Indicators\Volumes.mqh>
```

Jerarquía de herencia

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMFI

Métodos de la clase

Atributos	
MaPeriod	Devuelve el periodo de promediación
Applied	Devuelve el tipo de volumen a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Devuelve el periodo de promediación.

```
int  MaPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de volumen a aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor devuelto

Tipo de volumen a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string                symbol,         // Símbolo
   ENUM_TIMEFRAMES       period,         // Periodo
   int                   ma_period,      // Periodo de promediación
   ENUM_APPLIED_VOLUME   applied         // Tipo de volumen a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

ma_period

[in] Periodo de promediación.

applied

[in] Tipo de volumen a aplicar (enumeración ENUM_APPLIED_VOLUME).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_MFI es el de CiMFI).

© 2000-2025, MetaQuotes Ltd.

CiOBV

La clase CiOBV está diseñada para utilizar el indicador técnico Balance de Volúmenes (On Balance Volume, OBV).

Descripción

La clase CiOBV proporciona métodos para la creación, configuración y acceso a los datos del indicador Balance de Volúmenes (On Balance Volume, OBV).

Declaración

```
class CiOBV: public CIndicator
```

Título

```
#include <Indicators\Volumes.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiOBV

Métodos de la clase

Atributos	
Applied	Devuelve el tipo de volumen a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de volumen a aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor devuelto

Tipo de volumen a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string                symbol,      // Símbolo
   ENUM_TIMEFRAMES       period,      // Período
   ENUM_APPLIED_VOLUME   applied      // Tipo de volumen a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

applied

[in] Tipo de volumen a aplicar (enumeración ENUM_APPLIED_VOLUME).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_OBV para CiOBV).

© 2000-2025, MetaQuotes Ltd.

CiVolumes

La clase CiVolumes está diseñada para utilizar el indicador técnico Volúmenes.

Descripción

La clase CiVolumes proporciona métodos para la creación, configuración y acceso a los datos del indicador Volúmenes.

Declaración

```
class CiVolumes: public CIndicator
```

Título

```
#include <Indicators\Volumes.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiVolumes

Métodos de la clase

Atributos	
Applied	Devuelve el tipo de volumen a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de volumen a aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor devuelto

Tipo de volumen a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string                symbol,      // Símbolo
   ENUM_TIMEFRAMES       period,      // Período
   ENUM_APPLIED_VOLUME   applied      // Tipo de volumen a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

applied

[in] Tipo de volumen a aplicar (enumeración ENUM_APPLIED_VOLUME).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_VOLUMES es el de CiVolumes).

© 2000-2025, MetaQuotes Ltd.

Indicadores de Bill Williams

Los siguientes capítulos explican las características técnicas de las clases de la librería estándar MQL5 correspondientes al indicador de Bill Williams, y describe sus componentes clave.

Clase/grupo	Descripción
CiAC	Acelerador Oscilador
CiAlligator	Alligator
CiAO	Oscilador Asombroso
CiFractals	Fractales
CiGator	Oscilador Gator
CiBWMFI	Índice de Facilitación del Mercado

© 2000-2025, MetaQuotes Ltd.

CiAC

La clase CiAC está diseñada para utilizar el indicador técnico Oscilador Acelerador (Accelerator Oscillator).

Descripción

La clase CiAC proporciona métodos para la creación, configuración y acceso a los datos del indicador Accelerator Oscillator.

Declaración

```
class CiAC: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAC

Métodos de la clase

Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,      // Símbolo
   ENUM_TIMEFRAMES  period       // Periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_AC es el de CiAC).

© 2000-2025, MetaQuotes Ltd.

CiAlligator

La clase CiAlligator está diseñada para utilizar el indicador técnico Alligator.

Descripción

La clase CiAlligator proporciona métodos para la creación, configuración y acceso a los datos del indicador Alligator.

Declaración

```
class CiAlligator: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAlligator

Métodos de la clase

Atributos	
JawPeriod	Devuelve el periodo de promediación de la línea Mandíbula (Jaws)
JawShift	Devuelve el desplazamiento horizontal de la línea de la mandíbula
TeethPeriod	Devuelve el periodo de promediación de la línea Dientes (Teeths)
TeethShift	Devuelve el desplazamiento horizontal de la línea de los dientes
LipsPeriod	Devuelve el periodo de promediación de la línea Labios (Lips)
LipsShift	Devuelve el desplazamiento horizontal de la línea de los dientes
MaMethod	Devuelve el método de promediación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	

© 2000-2025, MetaQuotes Ltd.

Atributos	
Jaw	Devuelve el elemento del búfer de la línea Mandíbula (Jaws)
Teeth	Devuelve el elemento del búfer de la línea Dientes (Teeths)
Lips	Devuelve el elemento del búfer de la línea Labios (Lips)
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

JawPeriod

Devuelve el periodo de promediación de la línea de la mandíbula.

```
int  JawPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la línea de la mandíbula, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

JawShift

Devuelve el desplazamiento horizontal de la línea de la mandíbula.

```
int  JawShift() const
```

Valor devuelto

Desplazamiento horizontal de la línea de la mandíbula, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

TeethPeriod

Devuelve el periodo de promediación de la línea de los dientes (Teeth).

```
int  TeethPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la línea de los dientes (Teeth), definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

TeethShift

Devuelve el desplazamiento horizontal de la línea de los dientes (Teeth).

```
int  TeethShift() const
```

Valor devuelto

Desplazamiento horizontal de la línea de los dientes, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

LipsPeriod

Devuelve el periodo de promediación de la línea de los labios (Lips).

```
int  LipsPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la línea de los labios, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

LipsShift

Devuelve el desplazamiento horizontal de la línea de los labios (Lips).

```
int  LipsShift() const
```

Valor devuelto

Desplazamiento horizontal de la línea de los labios, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Devuelve el método de promediación.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor devuelto

Devuelve el método de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,           // Símbolo
   ENUM_TIMEFRAMES  period,           // Período
   int              jaw_period,       // Período de la línea de la mandíbula (Jaws)
   int              jaw_shift,        // Desplazamiento de la línea de la mandíbula (J
   int              teeth_period,     // Período de la línea de los dientes (Teeth)
   int              teeth_shift,      // Desplazamiento de la línea de los dientes (Te
   int              lips_period,      // Período de la línea de los labios (Lips)
   int              lips_shift,       // Desplazamiento de la línea de los labios (Lip
   ENUM_MA_METHOD   ma_method,        // Método de promediación
   int              applied           // Tipo de volumen a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

jaw_period

[in] Período de la línea de la mandíbula.

jaw_shift

[in] Desplazamiento de la línea de la mandíbula.

teeth_period

[in] Período de la línea de los dientes.

teeth_shift

[in] Desplazamiento de la línea de los dientes.

lips_period

[in] Período de la línea de los labios.

lips_shift

[in] Desplazamiento de la línea de los labios.

ma_method

[in] Método de la media móvil (enumeración ENUM_MA_METHOD).

applied

[in] Tipo de volumen a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Jaw

Devuelve el elemento del búfer de la línea de la mandíbula (Jaws) por el índice especificado.

```
double   Jaw(
   int   index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea de la mandíbula (Jaws) del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Teeth

Devuelve el elemento del búfer de la línea de los dientes (Teeth) por el índice especificado.

```
double  Teeth(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea de los dientes del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Lips

Devuelve el elemento del búfer de la línea de los labios (Lips) por el índice especificado.

```
double  Lips(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer de la línea de los labios del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_ALLIGATOR es el de CiAlligator).

© 2000-2025, MetaQuotes Ltd.

CiAO

La clase CiAO está diseñada para utilizar el indicador técnico Awesome Oscillator (Oscilador Impresionante).

Descripción

La clase CiAO proporciona métodos para la creación, configuración y acceso a los datos del indicador Awesome Oscillator.

Declaración

```
class CiAO: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAO

Métodos de la clase

Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,      // Símbolo
   ENUM_TIMEFRAMES  period       // Periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_AO es el de CiAO).

© 2000-2025, MetaQuotes Ltd.

CiFractals

La clase CiFractals está diseñada para utilizar el indicador técnico Fractals (Fractales).

Descripción

La clase CiFractals proporciona métodos para la creación, configuración y acceso a los datos del indicador Fractals.

Declaración

```
class CiFractals: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiFractals

Métodos de la clase

Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Upper	Devuelve el elemento del búfer superior
Lower	Devuelve el elemento del búfer inferior
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,      // Símbolo
   ENUM_TIMEFRAMES  period       // Periodo
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Upper

Devuelve el elemento del búfer superior por el índice especificado.

```
double  Upper(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer superior del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Lower

Devuelve el elemento del búfer inferior por el índice especificado.

```
double  Lower(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer inferior del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_FRACTALS es el de CiFractals).

© 2000-2025, MetaQuotes Ltd.

CiGator

La clase CiGator está diseñada para utilizar el indicador técnico Gator Oscillator (Oscilador Gator).

Descripción

La clase CiGator proporciona métodos para la creación, configuración y acceso a los datos del indicador Gator Oscillator.

Declaración

```
class CiGator: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiGator

Métodos de la clase

Atributos	
JawPeriod	Devuelve el periodo de promediación de la línea Mandíbula (Jaws)
JawShift	Devuelve el desplazamiento horizontal de la línea de la mandíbula
TeethPeriod	Devuelve el periodo de promediación de la línea Dientes (Teeths)
TeethShift	Devuelve el desplazamiento horizontal de la línea de los dientes
LipsPeriod	Devuelve el periodo de promediación de la línea Labios (Lips)
LipsShift	Devuelve el desplazamiento horizontal de la línea de los dientes
MaMethod	Devuelve el método de promediación
Applied	Devuelve el tipo de precio a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	

© 2000-2025, MetaQuotes Ltd.

Atributos	
Upper	Devuelve el elemento del búfer superior
Lower	Devuelve el elemento del búfer inferior
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

JawPeriod

Devuelve el período de promediación de la línea de la mandíbula (Jaws).

```
int  JawPeriod() const
```

Valor devuelto

Devuelve el período de promediación de la línea de la mandíbula, definido en la creación del indicador técnico.

© 2000-2025, MetaQuotes Ltd.

JawShift

Devuelve el desplazamiento horizontal de la línea de la mandíbula.

```
int  JawShift() const
```

Valor devuelto

Desplazamiento horizontal de la línea de la mandíbula, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

TeethPeriod

Devuelve el periodo de promediación de la línea de los dientes (Teeth).

```
int  TeethPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la línea de los dientes (Teeth), definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

TeethShift

Devuelve el desplazamiento horizontal de la línea de los dientes (Teeth).

```
int  TeethShift() const
```

Valor devuelto

Desplazamiento horizontal de la línea de los dientes, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

LipsPeriod

Devuelve el periodo de promediación de la línea de los labios (Lips).

```
int  LipsPeriod() const
```

Valor devuelto

Devuelve el periodo de promediación de la línea de los labios, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

LipsShift

Devuelve el desplazamiento horizontal de la línea de los labios (Lips).

```
int  LipsShift() const
```

Valor devuelto

Desplazamiento horizontal de la línea de los labios, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Devuelve el método de promediación.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor devuelto

Devuelve el método de promediación, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de precio o manejador a aplicar.

```
int  Applied() const
```

Valor devuelto

Tipo de precio o manejador a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string           symbol,           // Símbolo
   ENUM_TIMEFRAMES  period,           // Período
   int              jaw_period,       // Período de la línea de la mandíbula (Jaws)
   int              jaw_shift,        // Desplazamiento de la línea de la mandíbula (J
   int              teeth_period,     // Período de la línea de los dientes (Teeth)
   int              teeth_shift,      // Desplazamiento de la línea de los dientes (Te
   int              lips_period,      // Período de la línea de los labios (Lips)
   int              lips_shift,       // Desplazamiento de la línea de los labios (Lip
   ENUM_MA_METHOD   ma_method,        // Método de promediación
   int              applied           // Tipo de volumen a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

jaw_period

[in] Período de la línea de la mandíbula.

jaw_shift

[in] Desplazamiento de la línea de la mandíbula.

teeth_period

[in] Período de la línea de los dientes.

teeth_shift

[in] Desplazamiento de la línea de los dientes.

lips_period

[in] Período de la línea de los labios.

lips_shift

[in] Desplazamiento de la línea de los labios.

ma_method

[in] Método de promediación (enumeración ENUM_MA_METHOD).

applied

[in] Tipo de volumen a aplicar.

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Upper

Devuelve el elemento del búfer superior por el índice especificado.

```
double  Upper(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer superior del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Lower

Devuelve el elemento del búfer inferior por el índice especificado.

```
double  Upper(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

El elemento del búfer inferior del índice especificado, o EMPTY_VALUE si los datos no son correctos.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_GATOR para CiGator).

© 2000-2025, MetaQuotes Ltd.

CiBWMFI

La clase CiBWMFI está diseñada para utilizar el indicador técnico Índice de Facilitación del Mercado de Bill Williams.

Descripción

La clase CiBWMFI proporciona métodos para la creación, configuración y acceso a los datos del indicador Índice de Facilitación del Mercado de Bill Williams.

Declaración

```
class CiBWMFI: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Jerarquía de herencia

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBWMFI

Métodos de la clase

Atributos	
Applied	Devuelve el tipo de volumen a aplicar
Métodos de creación	
Create	Crea el indicador
Métodos de acceso a los datos	
Main	Devuelve el elemento de búfer
Entrada/salida	
virtual Type	Devuelve el identificador de tipo del objeto

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Métodos heredados de la clase CArray

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos heredados de la clase CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos heredados de la clase CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos heredados de la clase CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Applied

Devuelve el tipo de volumen a aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor devuelto

Tipo de volumen a aplicar, definido en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el indicador con los parámetros especificados. Utiliza Refresh() y GetData() para actualizar y obtener los valores del indicador.

```
bool  Create(
   string                symbol,      // Símbolo
   ENUM_TIMEFRAMES       period,      // Período
   ENUM_APPLIED_VOLUME   applied      // Tipo de volumen a aplicar
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

applied

[in] Tipo de volumen a aplicar (enumeración ENUM_APPLIED_VOLUME).

Valor devuelto

true si se ejecuta correctamente, false si el indicador no se puede crear.

© 2000-2025, MetaQuotes Ltd.

Main

Devuelve el elemento del búfer por el índice especificado.

```
double  Main(
   int  index      // Índice
   )
```

Parámetros

index

[in] Índice del elemento.

Valor devuelto

Elemento del búfer especificado en el índice, si se ejecuta correctamente; si los datos no son correctos devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_BWMFI es el de CiBWMFI).

© 2000-2025, MetaQuotes Ltd.

CiCustom

La clase CiCustom está diseñada para utilizar indicadores técnicos personalizados.

Descripción

La clase CiCustom proporciona métodos para la creación, configuración y acceso a los datos del indicador personalizado.

Declaración

```
class CiCustom: public CIndicator
```

Título

```
#include <Indicators\Custom.mqh>
```

Métodos de la clase

Atributos	
NumBuffers	Establece el número de búferes
NumParams	Obtiene el número de parámetros
ParamType	Obtiene el tipo del parámetro especificado
ParamLong	Obtiene el valor del parámetro especificado de tipo integer
ParamDouble	Obtiene el valor del parámetro especificado de tipo double
ParamString	Obtiene el valor del parámetro especificado de tipo string
Entrada/salida	
virtual Type	Obtiene el identificador de tipo del objeto

© 2000-2025, MetaQuotes Ltd.

NumBuffers

Establece el número de búferes.

```
bool  NumBuffers(
   int  buffers      // número de búferes
   )
```

Valor devuelto

true si se ejecuta correctamente, false si los búferes no se establecen.

© 2000-2025, MetaQuotes Ltd.

NumParams

Obtiene el número de parámetros.

```
int  NumParams() const
```

Valor devuelto

Número de parámetros utilizados en la creación del indicador.

© 2000-2025, MetaQuotes Ltd.

ParamType

Obtiene el tipo del parámetro especificado en el índice.

```
ENUM_DATATYPE  ParamType(
   int  index      // índice del parámetro
   ) const
```

Parámetros

index

[in] Índice del parámetro.

Valor devuelto

Devuelve el tipo de dato (valor de la enumeración ENUM_DATATYPE) del parámetro especificado en el índice, utilizado en la creación del indicador.

Nota

Si el índice del parámetro no es válido, devuelve WRONG_VALUE.

© 2000-2025, MetaQuotes Ltd.

ParamLong

Obtiene el valor del parámetro especificado de tipo long.

```
long  ParamLong(
   int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del parámetro.

Valor devuelto

El valor del parámetro especificado de tipo long, utilizado en la creación del indicador.

Nota

Si el número no es válido devuelve 0.

© 2000-2025, MetaQuotes Ltd.

ParamDouble

Obtiene el valor del parámetro especificado de tipo double.

```
double  ParamDouble(
   int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del parámetro.

Valor devuelto

El valor del parámetro especificado de tipo double, utilizado en la creación del indicador.

Nota

Si el número no es válido devuelve EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

ParamString

Obtiene el valor del parámetro especificado de tipo string.

```
string  ParamString(
   int  index      // índice
   ) const
```

Parámetros

index

[in] Índice del parámetro.

Valor devuelto

El valor del parámetro especificado de tipo string, utilizado en la creación del indicador.

Nota

Si el número no es válido devuelve una cadena vacía.

© 2000-2025, MetaQuotes Ltd.

Type

Devuelve el identificador de tipo del objeto.

```
virtual int  Type() const
```

Valor devuelto

Identificador de tipo del objeto (IND_CUSTOM es el de CiCustom).

© 2000-2025, MetaQuotes Ltd.

Clases de trading

Esta sección contiene detalles técnicos de las clases de trading, así como una descripción de los componentes relevantes de la Librería Estándar de MQL5.

Estas clases le ayudarán a ahorrar tiempo en el momento de crear sus propios programas (Asesores Expertos).

Las clases de trading de la Librería Estándar MQL5 se encuentran en el directorio de trabajo Include\Trade.

Clase/Grupo	Descripción
CAccountInfo	Clase de las propiedades de la cuenta de trading
CSymbolInfo	Clase de las propiedades del instrumento de trading
COrderInfo	Clase de las propiedades de las órdenes pendientes
CHistoryOrderInfo	Clase de las propiedades de las órdenes del historial
CPositionInfo	Clase de las propiedades de las posiciones abiertas
CDealInfo	Clase de las propiedades del historial de transacciones
CTrade	Clase de las ejecuciones de las operaciones de trading
CTerminalInfo	Clase para obtener las propiedades del entorno del programa mql5

© 2000-2025, MetaQuotes Ltd.

CAccountInfo

La clase CAccountInfo facilita el acceso a las propiedades de la cuenta de trading abierta.

Descripción

La clase CAccountInfo proporciona acceso a las propiedades de la cuenta de trading abierta.

Declaración

```
class CAccountInfo : public CObject
```

Título

```
#include <Trade\AccountInfo.mqh>
```

Jerarquía de herencia

CObject

CAccountInfo

Métodos de la clase por grupos

Acceso a las propiedades de tipo entero	
Login	Obtiene el número de la cuenta
TradeMode	Obtiene el modo de trading
TradeModeDescription	Obtiene el modo de trading en formato cadena
Leverage	Obtiene la cantidad de apalancamiento dado
StopoutMode	Obtiene el modo de stop out de la cuenta
StopoutModeDescription	Obtiene la descripción del modo de stop out de la cuenta
TradeAllowed	Obtiene la bandera de margen de trading
TradeExpert	Obtiene la bandera de margen de trading automático
LimitOrders	Obtiene el máximo número permitido de órdenes pendientes
MarginMode	Gets margin calculation mode
StopoutModeDescription	Gets margin calculation mode as a string
Acceso a las propiedades de tipo double	
Balance	Obtiene el saldo de la cuenta
Credit	Obtiene la cantidad de crédito
Profit	Obtiene el beneficio actual de la cuenta

© 2000-2025, MetaQuotes Ltd.

Acceso a las propiedades de tipo entero	
Equity	Obtiene el capital actual de la cuenta
Margin	Obtiene la cantidad de margen reservado
FreeMargin	Obtiene la cantidad de margen libre
MarginLevel	Obtiene el nivel de margen
MarginCall	Obtiene el nivel de margen del depósito
MarginStopOut	Obtiene el nivel de margen de Stop Out
Acceso a las propiedades de texto	
Name	Obtiene el nombre del cliente
Server	Obtiene el nombre del servidor de trading
Currency	Obtiene el nombre de la divisa del depósito
Company	Obtiene el nombre de la empresa que sirve la cuenta
Acceso a las funciones del API de MQL5	
InfoInteger	Obtiene el valor de la propiedad de tipo entero especificada
InfoDouble	Obtiene el valor de la propiedad de tipo double especificada
InfoString	Obtiene el valor de la propiedad de tipo string especificada
Métodos adicionales	
OrderProfitCheck	Obtiene el beneficio evaluado, basándose en los parámetros pasados
MarginCheck	Obtiene la cantidad de margen, necesaria para ejecutar la operación de trading
FreeMarginCheck	Obtiene la cantidad de margen libre disponible después de la ejecución de la operación de trading
MaxLotCheck	Obtiene el volumen máximo posible de la operación de trading

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Login

Obtiene el número de la cuenta.

```
long  Login() const
```

Valor devuelto

Número de la cuenta.

© 2000-2025, MetaQuotes Ltd.

TradeMode

Obtiene el modo de trading.

```
ENUM_ACCOUNT_TRADE_MODE  TradeMode() const
```

Valor devuelto

Modo de trading (valor de la enumeración ENUM_ACCOUNT_TRADE_MODE).

© 2000-2025, MetaQuotes Ltd.

TradeModeDescription

Obtiene el modo de trading en formato cadena.

```
string  TradeModeDescription() const
```

Valor devuelto

Modo de trading en formato cadena.

© 2000-2025, MetaQuotes Ltd.

Leverage

Obtiene la cantidad de apalancamiento dado.

```
long  Leverage() const
```

Valor devuelto

Cantidad de apalancamiento dado.

© 2000-2025, MetaQuotes Ltd.

StopoutMode

Obtiene el modo de Stop Out de la cuenta.

```
ENUM_ACCOUNT_STOPOUT_MODE  StopoutMode() const
```

Valor devuelto

Modo de Stop Out de la cuenta (valor de la enumeración ENUM_ACCOUNT_STOPOUT_MODE).

© 2000-2025, MetaQuotes Ltd.

StopoutModeDescription

Obtiene el modo de establecimiento de margen mínimo en formato cadena.

```
string  StopoutModeDescription() const
```

Valor devuelto

Modo de establecimiento de margen mínimo en formato cadena.

© 2000-2025, MetaQuotes Ltd.

MarginMode

Obtiene el modo de cálculo del margen.

```
ENUM_ACCOUNT_MARGIN_MODE  MarginMode() const
```

Valor devuelto

Modo de cálculo del margen de la enumeración ENUM_ACCOUNT_MARGIN_MODE.

© 2000-2025, MetaQuotes Ltd.

MarginModeDescription

Obtenemos el modo de cálculo del margen como línea.

```
string  MarginModeDescription() const
```

Valor devuelto

Modo de cálculo del margen como línea.

© 2000-2025, MetaQuotes Ltd.

TradeAllowed

Obtiene la bandera de margen de trading.

```
bool  TradeAllowed() const
```

Valor devuelto

Bandera de margen de trading.

© 2000-2025, MetaQuotes Ltd.

TradeExpert

Obtiene la bandera de margen de trading automático.

```
bool  TradeExpert() const
```

Valor devuelto

Bandera de margen de trading automático.

© 2000-2025, MetaQuotes Ltd.

LimitOrders

Obtiene el máximo número permitido de órdenes pendientes

```
int  LimitOrders() const
```

Valor devuelto

Obtiene el máximo número permitido de órdenes pendientes.

Nota

0 - sin límite.

© 2000-2025, MetaQuotes Ltd.

Balance

Obtiene el saldo de la cuenta.

```
double  Balance() const
```

Valor devuelto

El saldo de la cuenta (en la divisa del depósito).

© 2000-2025, MetaQuotes Ltd.

Credit

Obtiene la cantidad de crédito.

```
double  Credit() const
```

Valor devuelto

Cantidad de crédito (en la divisa del depósito).

© 2000-2025, MetaQuotes Ltd.

Profit

Obtiene el beneficio actual de la cuenta.

```
double  Profit() const
```

Valor devuelto

Beneficio actual de la cuenta (en la divisa del depósito).

© 2000-2025, MetaQuotes Ltd.

Equity

Obtiene el capital actual de la cuenta.

```
double  Equity() const
```

Valor devuelto

El capital actual de la cuenta (en la divisa del depósito).

© 2000-2025, MetaQuotes Ltd.

Margin

Obtiene la cantidad de margen reservado.

```
double  Margin() const
```

Valor devuelto

Cantidad de margen reservado (en la divisa del depósito).

© 2000-2025, MetaQuotes Ltd.

FreeMargin

Obtiene la cantidad de margen libre.

```
double  FreeMargin() const
```

Valor devuelto

Cantidad de margen libre (en la divisa del depósito).

© 2000-2025, MetaQuotes Ltd.

MarginLevel

Obtiene el nivel de margen.

```
double  MarginLevel() const
```

Valor devuelto

Nivel de margen.

© 2000-2025, MetaQuotes Ltd.

MarginCall

Obtiene el nivel de margen del depósito.

```
double  MarginCall() const
```

Valor devuelto

Nivel de margen del depósito.

© 2000-2025, MetaQuotes Ltd.

MarginStopOut

Obtiene el nivel de margen de Stop Out.

```
double  MarginStopOut() const
```

Valor devuelto

Nivel de margen de Stop Out.

© 2000-2025, MetaQuotes Ltd.

Name

Obtiene el nombre del cliente.

```
string  Name() const
```

Valor devuelto

Nombre del cliente.

© 2000-2025, MetaQuotes Ltd.

Server

Obtiene el nombre del servidor de trading.

```
string  Server() const
```

Valor devuelto

El nombre del servidor de trading.

© 2000-2025, MetaQuotes Ltd.

Currency

Obtiene el nombre de la divisa del depósito.

```
string  Currency() const
```

Valor devuelto

Nombre de la divisa del depósito.

© 2000-2025, MetaQuotes Ltd.

Company

Obtiene el nombre de la compañía que sirve la cuenta.

```
string  Company() const
```

Valor devuelto

Nombre de la compañía que sirve la cuenta.

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Obtiene el valor de la propiedad de tipo entero especificada.

```
long  InfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  prop_id     // ID de la propiedad
   ) const
```

Parámetros

prop_id

[in] Identificador de la propiedad. El valor puede ser cualquiera de los de la enumeración ENUM_ACCOUNT_INFO_INTEGER .

Valor devuelto

Valor de tipo long.

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Obtiene el valor de la propiedad de tipo double especificada.

```
double  InfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  prop_id     // ID de la propiedad
   ) const
```

Parámetros

prop_id

[in] Identificador de la propiedad. El valor puede ser cualquiera de los disponibles en la enumeración ENUM_ACCOUNT_INFO_DOUBLE .

Valor devuelto

Valor de tipo double.

© 2000-2025, MetaQuotes Ltd.

InfoString

Obtiene el valor de la propiedad de tipo string especificada.

```
string  InfoString(
   ENUM_ACCOUNT_INFO_STRING  prop_id     // ID de la propiedad
   ) const
```

Parámetros

prop_id

[in] Identificador de la propiedad. El valor puede ser cualquiera de los disponibles en la enumeración ENUM_ACCOUNT_INFO_STRING .

Valor devuelto

Valor de tipo string.

© 2000-2025, MetaQuotes Ltd.

OrderProfitCheck

La función calcula el beneficio de la cuenta actual, basándose en los parámetros pasados. Esta función se utiliza para evaluar previamente el resultado de una operación de trading. El valor se devuelve en la divisa de la cuenta.

```
double  OrderProfitCheck(
   const string        symbol,              // símbolo
   ENUM_ORDER_TYPE     trade_operation,     // tipo de operación (ORDER_TYPE_BUY or OR
   double              volume,              // volumen
   double              price_open,          // precio de apertura
   double              price_close          // precio de cierre
   ) const
```

Parámetros

symbol

[in] Símbolo de la operación de trading.

trade_operation

[in] Tipo de operación de trading (enumeración ENUM_ORDER_TYPE).

volume

[in] Volumen de la operación.

price_open

[in] Precio de apertura.

price_close

[in] Precio de cierre.

Valor devuelto

Si se ejecuta correctamente devuelve el beneficio, o EMPTY_VALUE en caso de error.

© 2000-2025, MetaQuotes Ltd.

MarginCheck

Obtiene la cantidad de margen, necesaria para ejecutar la operación comercial.

```
double  MarginCheck(
   const string        symbol,              // símbolo
   ENUM_ORDER_TYPE     trade_operation,     // operación
   double              volume,              // volumen
   double              price                // precio
   ) const
```

Parámetros

symbol

[in] Símbolo de la operación de trading.

trade_operation

[in] Tipo de operación (enumeración ENUM_ORDER_TYPE).

volume

[in] Volumen de la operación.

price

[in] Precio de la operación.

Valor devuelto

Cantidad de margen, necesaria para ejecutar la operación comercial.

© 2000-2025, MetaQuotes Ltd.

FreeMarginCheck

Obtiene la cantidad de margen libre disponible después de la ejecución de la operación comercial.

```
double  FreeMarginCheck(
   const string        symbol,              // símbolo
   ENUM_ORDER_TYPE     trade_operation,     // operación
   double              volume,              // volumen
   double              price                // precio
   ) const
```

Parámetros

symbol

[in] Símbolo de la operación de trading.

trade_operation

[in] Tipo de operación (enumeración ENUM_ORDER_TYPE).

volume

[in] Volumen de la operación.

price

[in] Precio de la operación.

Valor devuelto

Cantidad de margen libre disponible después de la ejecución de la operación comercial.

© 2000-2025, MetaQuotes Ltd.

MaxLotCheck

Obtiene el volumen máximo posible de la operación de trading.

```
double  MaxLotCheck(
   const string         symbol,              // símbolo
   ENUM_ORDER_TYPE      trade_operation,     // operación
   double               price,               // precio
   double               percent=100          // porcentaje de margen disponible (0-100%
   ) const
```

Parámetros

symbol

[in] Símbolo de la operación de trading.

trade_operation

[in] Tipo de operación de trading (enumeración ENUM_ORDER_TYPE).

price

[in] Precio de la operación.

percent=100

[in] Porcentaje de margen disponible (0-100%), utilizado en la operación de trading.

Valor devuelto

Volumen máximo posible de la operación de trading.

© 2000-2025, MetaQuotes Ltd.

CSymbolInfo

La clase CSymbolInfo facilita el acceso a las propiedades del símbolo.

Descripción

La clase CSymbolInfo proporciona acceso a las propiedades del símbolo.

Declaración

```
class CSymbolInfo : public CObject
```

Título

```
#include <Trade\SymbolInfo.mqh>
```

Jerarquía de herencia

CObject

CSymbolInfo

Métodos de la clase por grupos

Control	
Refresh	Actualiza los datos del símbolo
RefreshRates	Actualiza las cotizaciones del símbolo
Propiedades	
Name	Obtiene/establece el nombre del símbolo
Select	Obtiene/establece la bandera del símbolo "Observación del Mercado"
IsSynchronized	Comprueba la sincronización del símbolo con el servidor
Volúmenes	
Volume	Obtiene el volumen de la última transacción
VolumeHigh	Obtiene el volumen máximo del día
VolumeLow	Obtiene el volumen mínimo del día
Varios	
Time	Obtiene la hora de la última cotización
Spread	Obtiene la cantidad de spread (en puntos)
SpreadFloat	Obtiene la bandera del spread flotante
TicksBookDepth	Obtiene la profundidad de los ticks
Niveles	

© 2000-2025, MetaQuotes Ltd.

Control	
StopsLevel	Obtiene el nivel de stop mínimo de las órdenes (en puntos)
FreezeLevel	Obtiene la distancia de congelación de las operaciones de trading (en puntos)
Precios Bid	
Bid	Obtiene el precio Bid actual
BidHigh	Obtiene el precio Bid máximo del día
BidLow	Obtiene el precio Bid mínimo del día
Precios Ask	
Ask	Obtiene el precio Ask actual
AskHigh	Obtiene el precio Ask máximo del día
AskLow	Obtiene el precio Ask mínimo del día
Precios	
Last	Obtiene el precio Last actual
LastHigh	Obtiene el precio Last máximo del día
LastLow	Obtiene el precio Last mínimo del día
Modos de trading	
TradeCalcMode	Obtiene el modo de cálculo de coste del contrato
TradeCalcModeDescription	Obtiene el modo de cálculo de coste del contrato en formato string
TradeMode	Obtiene el tipo de ejecución de la orden
TradeModeDescription	Obtiene el tipo de ejecución de la orden en formato string
TradeExecution	Obtiene el modo de cierre de las transacciones
TradeExecutionDescription	Obtiene el modo de cierre de las transacciones en formato string
Swaps	
SwapMode	Obtiene el modelo de cálculo de swap
SwapModeDescription	Obtiene el modelo de cálculo de swap en formato cadena
SwapRollover3days	Obtiene el día de cargo de swap triple
SwapRollover3daysDescription	Obtiene el día de cargo de swap triple en formato string
Márgenes y banderas	
MarginInitial	Obtiene el valor del margen inicial
MarginMaintenance	Obtiene el valor del margen de mantenimiento

© 2000-2025, MetaQuotes Ltd.

Control	
MarginLong	Obtiene la tasa de margen de carga de las posiciones largas
MarginShort	Obtiene la tasa de margen de carga de las posiciones cortas
MarginLimit	Obtiene la tasa de margen de carga de las órdenes Limit
MarginStop	Obtiene la tasa de margen de carga de las órdenes Stop
MarginStopLimit	Obtiene la tasa de margen de carga de las órdenes StopLimit
TradeTimeFlags	Obtiene las banderas de los modos de expiración de orden permitidos
TradeFillFlags	Obtiene las banderas de los modos de relleno de orden permitidos
Cuantificación	
Digits	Obtiene el número de dígitos después del punto
Point	Obtiene el valor de un punto
TickValue	Obtiene el costo del tick (el cargo mínimo del precio)
TickValueProfit	Obtiene el precio calculado del tick de una posición ganadora
TickValueLoss	Obtiene el precio calculado del tick de una posición perdedora
TickSize	Obtiene el cambio mínimo del precio
Tamaños del contrato	
ContractSize	Obtiene el tamaño del contrato
LotsMin	Obtiene el volumen mínimo para cerrar una transacción
LotsMax	Obtiene el volumen máximo para cerrar una transacción
LotsStep	Obtiene el paso mínimo de cambio de volumen para cerrar una transacción
LotsLimit	Obtiene el volumen máximo permitido de la posición abierta y de las órdenes pendientes (dirección insensible) del símbolo.
Tamaños de los swaps	
SwapLong	Obtiene el valor del swap de la posición larga
SwapShort	Obtiene el valor del swap de la posición corta
Propiedades de texto	
CurrencyBase	Obtiene el nombre de la divisa base del símbolo
CurrencyProfit	Obtiene el nombre de la divisa del beneficio
CurrencyMargin	Obtiene el nombre de la divisa del margen
Bank	Obtiene el nombre de la fuente de cotización actual

© 2000-2025, MetaQuotes Ltd.

Control	
Descripción	Obtiene la descripción string del símbolo
Path	Obtiene la ruta en el árbol de símbolos
Propiedades del símbolo	
SessionDeals	Obtiene el número de transacciones de la sesión actual
SessionBuyOrders	Obtiene el número actual de órdenes Buy
SessionSellOrders	Obtiene el número actual de órdenes Sell
SessionTurnover	Obtiene el resumen del volumen de la sesión actual
SessionInterest	Obtiene el resumen del interés abierto de la sesión actual
SessionBuyOrdersVolume	Obtiene el volumen actual de las órdenes de compra
SessionSellOrdersVolume	Obtiene el volumen actual de las órdenes de venta
SessionOpen	Obtiene el precio de apertura de la sesión actual
SessionClose	Obtiene el precio de cierre de la sesión actual
SessionAW	Obtiene el precio medio ponderado de la sesión actual
SessionPriceSettlement	Obtiene el precio de liquidación de la sesión actual
SessionPriceLimitMin	Obtiene el precio mínimo de la sesión actual
SessionPriceLimitMax	Obtiene el precio máximo de la sesión actual
Acceso a las funciones del API de MQL5	
InfoInteger	Obtiene el valor de la propiedad de tipo entero especificada
InfoDouble	Obtiene el valor de la propiedad de tipo double especificada
InfoString	Obtiene el valor de la propiedad de tipo string especificada
Funciones de servicio	
NormalizePrice	Devuelve el valor del precio, normalizado mediante las propiedades del símbolo

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza los datos del símbolo.

```
void  Refresh()
```

Valor devuelto

Ninguno.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

RefreshRates

Actualiza las cotizaciones del símbolo.

```
bool  RefreshRates()
```

Valor devuelto

true - en caso de éxito, false - si las cotizaciones no se pueden refrescar.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Name

Obtiene el nombre del símbolo.

```
string  Name() const
```

Valor devuelto

Nombre del símbolo.

Name

Establece el nombre del símbolo.

```
bool  Name(string name)
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Select

Obtiene la bandera del símbolo "Observación del Mercado".

```
bool  Select() const
```

Valor devuelto

Obtiene la bandera del símbolo "Observación del Mercado".

Select

Establece la bandera del símbolo "Observación del Mercado".

```
bool  Select()
```

Valor devuelto

true - en caso de éxito, false - si la bandera no se puede cambiar.

© 2000-2025, MetaQuotes Ltd.

IsSynchronized

Comprueba la sincronización del símbolo con el servidor.

```
bool  IsSynchronized() const
```

Valor devuelto

true - si el símbolo está sincronizado con el servidor, false - en caso contrario.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Volume

Obtiene el volumen de la última transacción.

```
long  Volume() const
```

Valor devuelto

Volumen de la última transacción.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

VolumeHigh

Obtiene el volumen máximo del día.

```
long  VolumeHigh() const
```

Valor devuelto

Volumen máximo del día.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

VolumeLow

Obtiene el volumen mínimo del día.

```
long  VolumeLow() const
```

Valor devuelto

Volumen mínimo del día.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Time

Obtiene la hora de la última cotización.

```
datetime  Time() const
```

Valor devuelto

Hora de la última cotización.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Spread

Obtiene la cantidad de spread (en puntos).

```
int  Spread() const
```

Valor devuelto

Obtiene la cantidad de spread (en puntos).

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SpreadFloat

Obtiene la bandera del spread flotante.

```
bool  SpreadFloat() const
```

Valor devuelto

Bandera del spread flotante.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TicksBookDepth

Obtiene la profundidad de los ticks.

```
int  TicksBookDepth() const
```

Valor devuelto

Profundidad de los ticks.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

StopsLevel

Obtiene el nivel de stop mínimo de las órdenes (en puntos).

```
int  StopsLevel() const
```

Valor devuelto

Nivel de stop mínimo de las órdenes (en puntos).

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

FreezeLevel

Obtiene el nivel de congelación (en puntos).

```
int  FreezeLevel() const
```

Valor devuelto

Distancia del nivel de congelación (en puntos).

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Bid

Obtiene el precio Bid actual.

```
double  Bid() const
```

Valor devuelto

Precio Bid actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

BidHigh

Obtiene el precio Bid máximo del día.

```
double  BidHigh() const
```

Valor devuelto

Precio Bid máximo del día.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

BidLow

Obtiene el precio Bid mínimo del día.

```
double  BidLow() const
```

Valor devuelto

Precio Bid mínimo del día.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Ask

Obtiene el precio Ask actual.

```
double  Ask() const
```

Valor devuelto

Precio Ask actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

AskHigh

Obtiene el precio Ask máximo del día.

```
double  AskHigh() const
```

Valor devuelto

Precio Ask máximo del día.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

AskLow

Obtiene el precio Ask mínimo del día.

```
double  AskLow() const
```

Valor devuelto

Precio Ask mínimo del día.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Last

Obtiene el precio Last actual.

```
double  Last() const
```

Valor devuelto

Precio Last actual.

© 2000-2025, MetaQuotes Ltd.

LastHigh

Obtiene el precio Last máximo del día.

```
double  LastHigh() const
```

Valor devuelto

Precio Last máximo del día.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

LastLow

Obtiene el precio Last mínimo del día.

```
double  LastLow() const
```

Valor devuelto

Precio Last mínimo del día.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TradeCalcMode

Obtiene el modo de cálculo de coste del contrato.

```
ENUM_SYMBOL_CALC_MODE  TradeCalcMode() const
```

Valor devuelto

Modo de cálculo de coste del contrato (valor de la enumeración ENUM_SYMBOL_CALC_MODE).

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TradeCalcModeDescription

Obtiene el modo de cálculo de coste del contrato en formato string.

```
string  TradeCalcModeDescription() const
```

Valor devuelto

Modo de cálculo de coste del contrato en formato string.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TradeMode

Obtiene el tipo de ejecución de la orden.

```
ENUM_SYMBOL_TRADE_MODE  TradeMode() const
```

Valor devuelto

Tipo de ejecución de la orden (valor de la enumeración ENUM_SYMBOL_TRADE_MODE).

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TradeModeDescription

Obtiene el modo de trading en formato cadena.

```
string  TradeModeDescription() const
```

Valor devuelto

Modo de trading en formato cadena.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TradeExecution

Obtiene el modo de ejecución de la transacción.

```
ENUM_SYMBOL_TRADE_EXECUTION  TradeExecution() const
```

Valor devuelto

Modo de ejecución de la transacción (valor de la enumeración ENUM_SYMBOL_TRADE_EXECUTION).

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TradeExecutionDescription

Obtiene la descripción del modo de ejecución de la transacción en formato string.

```
string  TradeExecutionDescription() const
```

Valor devuelto

Modo de ejecución de la transacción en formato string.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SwapMode

Obtiene el modelo de cálculo de swap.

```
ENUM_SYMBOL_SWAP_MODE  SwapMode() const
```

Valor devuelto

Modelo de cálculo de swap (valor de la enumeración ENUM_SYMBOL_SWAP_MODE).

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SwapModeDescription

Obtiene la descripción del modo de swap en formato cadena.

```
string  SwapModeDescription() const
```

Valor devuelto

Descripción del modo de swap en formato cadena.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SwapRollover3days

Obtiene el día de rollover swap.

```
ENUM_DAY_OF_WEEK  SwapRollover3days() const
```

Valor devuelto

Día de rollover swap (valor de la enumeración ENUM_DAY_OF_WEEK).

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SwapRollover3daysDescription

Obtiene el día de rollover swap en formato string.

```
string  SwapRollover3daysDescription() const
```

Valor devuelto

Día de rollover swap en formato string.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

MarginInitial

Obtiene el valor del margen inicial.

```
double  MarginInitial()
```

Valor devuelto

Valor del margen inicial.

Nota

Apunta a la cantidad de margen (en la divisa de margen del instrumento) que se carga de un lote. Se utiliza para comprobar el patrimonio del cliente cuando entra en el mercado.

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

MarginMaintenance

Obtiene el valor del margen de mantenimiento.

```
double  MarginMaintenance()
```

Valor devuelto

Valor del margen de mantenimiento.

Nota

Apunta a la cantidad de margen (en la divisa de margen del instrumento) que se carga de un lote. Se utiliza para comprobar el patrimonio del cliente cuando cambia el estado de la cuenta. Si el margen de mantenimiento es igual a 0, se utiliza el margen inicial.

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

MarginLong

Obtiene la tasa de margen de carga de las posiciones largas.

```
double  MarginLong() const
```

Valor devuelto

Tasa de margen de carga de las posiciones largas.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

MarginShort

Obtiene la tasa de margen de carga de las posiciones cortas.

```
double  MarginShort() const
```

Valor devuelto

Tasa de margen de carga de las posiciones cortas.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

MarginLimit

Obtiene la tasa de margen de carga de las órdenes Limit.

```
double  MarginLimit() const
```

Valor devuelto

Tasa de margen de carga de las órdenes Limit.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

MarginStop

Obtiene la tasa de margen de carga de las órdenes Stop.

```
double  MarginStop() const
```

Valor devuelto

Tasa de margen de carga de las órdenes Stop.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

MarginStopLimit

Obtiene la tasa de margen de carga de las órdenes Stop Limit.

```
double  MarginStopLimit() const
```

Valor devuelto

Tasa de margen de carga de las órdenes Stop Limit.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TradeTimeFlags

Obtiene las banderas de los modos de expiración de orden permitidos.

```
int  TradeTimeFlags() const
```

Valor devuelto

Banderas de los modos de expiración de orden permitidos.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TradeFillFlags

Obtiene las banderas de los modos de relleno de orden permitidos.

```
int  TradeFillFlags() const
```

Valor devuelto

Banderas de los modos de relleno de orden permitidos.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Digits

Obtiene el número de dígitos después del punto.

```
int  Digits() const
```

Valor devuelto

Obtiene el número de dígitos después del punto.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Point

Obtiene el valor del punto.

```
double  Point() const
```

Valor devuelto

Obtiene el valor del punto.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TickValue

Obtiene el coste del tick (cambio mínimo del precio).

```
double  TickValue() const
```

Valor devuelto

Coste del tick (cambio mínimo del precio).

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TickValueProfit

Obtiene el precio calculado del tick de una posición ganadora.

```
double  TickValueProfit() const
```

Valor devuelto

Precio calculado del tick de una posición ganadora.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TickValueLoss

Obtiene el precio calculado del tick de una posición perdedora.

```
double  TickValueLoss() const
```

Valor devuelto

Precio calculado del tick de una posición perdedora.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

TickSize

Obtiene el cambio mínimo del precio.

```
double  TickSize() const
```

Valor devuelto

Cambio mínimo del precio.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

ContractSize

Obtiene el tamaño del contrato.

```
double  ContractSize() const
```

Valor devuelto

Tamaño del contrato.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

LotsMin

Obtiene el volumen mínimo para cerrar una transacción.

```
double  LotsMin() const
```

Valor devuelto

Volumen mínimo para cerrar una transacción.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

LotsMax

Obtiene el volumen máximo para cerrar una transacción.

```
double  LotsMax() const
```

Valor devuelto

Volumen máximo para cerrar una transacción.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

LotsStep

Obtiene el paso mínimo de cambio de volumen para cerrar una transacción.

```
double  LotsStep() const
```

Valor devuelto

Paso mínimo de cambio de volumen para cerrar una transacción.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

LotsLimit

Obtiene el volumen máximo permitido de la posición abierta y de las órdenes pendientes (dirección insensible) del símbolo.

```
double  LotsLimit() const
```

Valor devuelto

Volumen máximo permitido de la posición abierta y de las órdenes pendientes (dirección insensible) del símbolo.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SwapLong

Obtiene el valor del swap de la posición larga.

```
double  SwapLong() const
```

Valor devuelto

Valor del swap de la posición larga.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SwapShort

Obtiene el valor del swap de la posición corta.

```
double  SwapShort() const
```

Valor devuelto

Valor del swap de la posición corta.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

CurrencyBase

Obtiene el nombre de la divisa base del símbolo.

```
string  CurrencyBase() const
```

Valor devuelto

Nombre de la divisa base del símbolo.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

CurrencyProfit

Obtiene el nombre de la divisa del beneficio.

```
string  CurrencyProfit() const
```

Valor devuelto

Nombre de la divisa del beneficio.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

CurrencyMargin

Obtiene el nombre de la divisa del margen.

```
string  CurrencyMargin() const
```

Valor devuelto

Nombre de la divisa del margen.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Bank

Obtiene el nombre de la fuente de cotización actual.

```
string  Bank() const
```

Valor devuelto

Nombre de la fuente de cotización actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Descripción

Obtiene la descripción string del símbolo.

```
string  Description() const
```

Valor devuelto

Descripción string del símbolo.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

Path

Obtiene la ruta en el árbol de símbolos.

```
string  Path() const
```

Valor devuelto

Obtiene la ruta en el árbol de símbolos.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionDeals

Obtiene el número de transacciones de la sesión actual.

```
long  SessionDeals() const
```

Valor devuelto

Número de transacciones de la sesión actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionBuyOrders

Obtiene el número actual de órdenes Buy.

```
long  SessionBuyOrders() const
```

Valor devuelto

Número actual de órdenes Buy.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionSellOrders

Obtiene el número actual de órdenes de venta.

```
long  SessionSellOrders() const
```

Valor devuelto

Número actual de órdenes de venta.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionTurnover

Obtiene el resumen del volumen de la sesión actual.

```
double  SessionTurnover() const
```

Valor devuelto

Resumen del volumen de la sesión actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionInterest

Obtiene el resumen del interés abierto de la sesión actual.

```
double  SessionInterest() const
```

Valor devuelto

Resumen del interés abierto de la sesión actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionBuyOrdersVolume

Obtiene el volumen actual de las órdenes de compra.

```
double  SessionBuyOrdersVolume() const
```

Valor devuelto

Volumen actual de las órdenes de compra.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionSellOrdersVolume

Obtiene el volumen actual de órdenes de venta.

```
double  SessionSellOrdersVolume() const
```

Valor devuelto

Volumen actual de órdenes de venta.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionOpen

Obtiene el precio de apertura de la sesión actual.

```
double  SessionOpen() const
```

Valor devuelto

Precio de apertura de la sesión actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionClose

Obtiene el precio de cierre de la sesión actual.

```
double  SessionClose() const
```

Valor devuelto

Precio de cierre de la sesión actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionAW

Obtiene el precio medio ponderado de la sesión actual.

```
double  SessionAW() const
```

Valor devuelto

Precio medio ponderado de la sesión actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionPriceSettlement

Obtiene el precio de liquidación de la sesión actual.

```
double  SessionPriceSettlement() const
```

Valor devuelto

Precio de liquidación de la sesión actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionPriceLimitMin

Obtiene el precio mínimo de la sesión actual.

```
double  SessionPriceLimitMin() const
```

Valor devuelto

Precio mínimo de la sesión actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

SessionPriceLimitMax

Obtiene el precio máximo de la sesión actual.

```
double  SessionPriceLimitMax() const
```

Valor devuelto

Precio máximo de la sesión actual.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Obtiene el valor de la propiedad de tipo entero especificada.

```
bool  InfoInteger(
   ENUM_SYMBOL_INFO_INTEGER  prop_id,      // ID de la propiedad
   long&                     var           // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de tipo integer (valor de la enumeración ENUM_SYMBOL_INFO_INTEGER).

var

[out] Referencia a la variable de tipo long, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Obtiene el valor de la propiedad de tipo double especificada.

```
bool  InfoDouble(
   ENUM_SYMBOL_INFO_DOUBLE  prop_id,      // ID de la propiedad
   double&                  var           // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de tipo double (valor de la enumeración ENUM_SYMBOL_INFO_DOUBLE).

var

[out] Referencia a la variable de tipo double para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

InfoString

Obtiene el valor de la propiedad de tipo string especificada.

```
bool  InfoString(
   ENUM_SYMBOL_INFO_STRING  prop_id,      // ID de la propiedad
   string&                  var           // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de texto.

var

[out] Referencia a la variable de tipo string, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

NormalizePrice

Devuelve el valor del precio, normalizado mediante las propiedades del símbolo.

```
double  NormalizePrice(
   double  price   // precio
   ) const
```

Parámetros

price

[in] Precio.

Valor devuelto

Precio normalizado.

Nota

El símbolo se tiene que seleccionar con el método Name.

© 2000-2025, MetaQuotes Ltd.

COrderInfo

La clase COrderInfo facilita el acceso a las propiedades de la orden pendiente.

Descripción

La clase COrderInfo proporciona acceso a las propiedades de la orden pendiente.

Declaración

```
class COrderInfo : public CObject
```

Título

```
#include <Trade\OrderInfo.mqh>
```

Jerarquía de herencia

CObject

COrderInfo

Métodos de la clase por grupos

Acceso a las propiedades de tipo entero	
Ticket	Obtiene el ticket de la orden, previamente seleccionado para el acceso
TimeSetup	Obtiene la hora de colocación de la orden
TimeSetupMsc	Recibe la hora de colocación de la orden en milisegundos desde el 01-01-1970
OrderType	Obtiene el tipo de orden
OrderTypeDescription	Obtiene el tipo de orden en formato string
State	Obtiene el estado de la orden
StateDescription	Obtiene el estado de la orden en formato string
TimeExpiration	Obtiene la hora de expiración de la orden
TimeDone	Obtiene la hora de ejecución o cancelación de la orden
TimeDoneMsc	Recibe la hora de ejecución o cancelación de la orden en milisegundos desde el 01-01-1970
TypeFilling	Obtiene el tipo de ejecución de la orden
TypeFillingDescription	Obtiene el tipo de ejecución de la orden en formato string
TypeTime	Obtiene el tipo de orden en la hora de expiración
TypeTimeDescription	Obtiene el tipo de orden por hora de expiración en formato string

© 2000-2025, MetaQuotes Ltd.

Acceso a las propiedades de tipo entero	
Magic	Obtiene el ID del experto que coloca la orden
PositionId	Obtiene el ID de la posición
Acceso a las propiedades de tipo double	
VolumeInitial	Obtiene el volumen inicial de la orden
VolumeCurrent	Obtiene el volumen vacío de la orden
PriceOpen	Obtiene el precio de la orden
StopLoss	Obtiene el Stop Loss de la orden
TakeProfit	Obtiene el Take Profit de la orden
PriceCurrent	Obtiene el precio actual por el símbolo de la orden
PriceStopLimi	Obtiene el precio de establecimiento de la orden limit
Acceso a las propiedades de texto	
Symbol	Obtiene el nombre del símbolo de la orden
Comment	Obtiene el comentario de la orden
Acceso a las funciones del API de MQL5	
InfoInteger	Obtiene el valor de la propiedad de tipo entero especificada
InfoDouble	Obtiene el valor de la propiedad de tipo double especificada
InfoString	Obtiene el valor de la propiedad de tipo string especificada
State	
StoreState	Guarda los parámetros de la orden
CheckState	Comprueba los parámetros actuales con los parámetros guardados
Selección	
Select	Selecciona la orden por ticket, para acceder posteriormente a sus propiedades
SelectByIndex	Selecciona la orden por índice, para acceder posteriormente a sus propiedades

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Ticket

Obtiene el ticket de la orden, previamente seleccionada con el método Select.

```
ulong  Ticket() const
```

Valor devuelto

El ticket de la orden, si se ejecuta correctamente; en caso contrario - ULONG_MAX.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeSetup

Obtiene la hora de colocación de la orden.

```
datetime  TimeSetup() const
```

Valor devuelto

Hora de colocación de la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeSetupMsc

Recibe la hora de colocación de la orden para su ejecución, en milisegundos desde el 01-01-1970.

```
ulong  TimeSetupMsc() const
```

Valor devuelto

Hora de colocación de la orden para su ejecución, en milisegundos desde el 01-01-1970.

Nota

La orden se tiene que seleccionar previamente con el método Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

OrderType

Obtiene el tipo de la orden.

```
ENUM_ORDER_TYPE  OrderType()
```

Valor devuelto

Tipo de orden (valor de la enumeración ENUM_ORDER_TYPE).

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeDescription

Obtiene el tipo de orden en formato string.

```
string  TypeDescription() const
```

Valor devuelto

Tipo de orden en formato string.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

State

Obtiene el estado de la orden.

```
ENUM_ORDER_STATE  State() const
```

Valor devuelto

Estado de la orden (valor de la enumeración ENUM_ORDER_STATE).

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

StateDescription

Obtiene el estado de la orden en formato string.

```
string  StateDescription() const
```

Valor devuelto

Estado de la orden en formato string.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeExpiration

Obtiene la hora de expiración de la orden.

```
datetime  TimeExpiration() const
```

Valor devuelto

Hora de expiración de la orden, establecida en su colocación.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeDone

Obtiene la hora de ejecución o cancelación.

```
datetime  TimeDone() const
```

Valor devuelto

Hora de ejecución o cancelación.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeDoneMsc

Recibe la hora de ejecución o cancelación de la orden en milisegundos desde el 01-01-1970.

```
ulong  TimeDoneMsc() const
```

Valor devuelto

Hora de ejecución o cancelación de la orden en milisegundos desde el 01-01-1970.

Nota

La orden se tiene que seleccionar previamente con el método Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeFilling

Obtiene el tipo de relleno de la orden.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Valor devuelto

Tipo de relleno de la orden (valor de la enumeración ENUM_ORDER_TYPE_FILLING).

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeFillingDescription

Obtiene el tipo de relleno de la orden en formato string.

```
string  TypeFillingDescription() const
```

Valor devuelto

Tipo de relleno de la orden en formato string.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeTime

Obtiene el tipo de orden en la hora de expiración.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Valor devuelto

Tipo de orden en la hora de expiración.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeTimeDescription

Obtiene el tipo de orden por hora de expiración en formato string.

```
string  TypeTimeDescription() const
```

Valor devuelto

Tipo de orden por hora de expiración en formato string.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Magic

Obtiene el ID del experto que coloca la orden.

```
long  Magic() const
```

Valor devuelto

ID del experto que coloca la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PositionId

Obtiene el ID de la posición.

```
long  PositionId() const
```

Valor devuelto

ID de la posición correspondiente a la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

VolumeInitial

Obtiene el volumen inicial de la orden.

```
double  VolumeInitial() const
```

Valor devuelto

Obtiene el volumen inicial de la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

VolumeCurrent

Obtiene el volumen vacío de la orden.

```
double  VolumeCurrent() const
```

Valor devuelto

Volumen vacío de la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PriceOpen

Obtiene el precio de la orden.

```
double  PriceOpen() const
```

Valor devuelto

Precio de colocación de la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

StopLoss

Obtiene el Stop Loss de la orden.

```
double  StopLoss() const
```

Valor devuelto

Stop Loss de la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TakeProfit

Obtiene el Take Profit de la orden.

```
double  TakeProfit() const
```

Valor devuelto

Take Profit de la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PriceCurrent

Obtiene el precio actual por símbolo de la orden.

```
double  PriceCurrent() const
```

Valor devuelto

Precio actual por símbolo de la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PriceStopLimi

Obtiene el precio de establecimiento limit order

```
double  PriceStopLimit() const
```

Valor devuelto

Precio de establecimiento limit order.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Symbol

Obtiene el nombre del símbolo de la orden.

```
string  Symbol() const
```

Valor devuelto

Nombre del símbolo de la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Comment

Obtiene el comentario de la orden.

```
string  Comment() const
```

Valor devuelto

Comentario de la orden.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Obtiene el valor de la propiedad de tipo entero especificada.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,     // ID de la propiedad
   long&                        var          // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de tipo integer (valor de la enumeración ENUM_ORDER_PROPERTY_INTEGER).

var

[out] Referencia a la variable de tipo long, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Obtiene el valor de la propiedad de tipo double especificada.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // ID de la propiedad
   double&                     var          // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de tipo double (valor de la enumeración ENUM_ORDER_PROPERTY_DOUBLE).

var

[out] Referencia a la variable de tipo double para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoString

Obtiene el valor de la propiedad de tipo string especificada.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // ID de la propiedad
   string&                     var           // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de texto.

var

[out] Referencia a la variable de tipo string, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La orden se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

StoreState

Guarda los parámetros de la orden.

```
void  StoreState()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

CheckState

Comprueba los parámetros actuales con los parámetros guardados.

```
bool  CheckState()
```

Valor devuelto

true - si los parámetros de la orden cambian desde la última llamada al método StoreState() , en caso contrario - false.

© 2000-2025, MetaQuotes Ltd.

Select

Selecciona una orden por ticket para acceder posteriormente a sus propiedades.

```
bool  Select(
   ulong  ticket   // ticket de la orden
   )
```

Valor devuelto

true - si se ejecuta correctamente, false - si la orden no se puede seleccionar.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

Selecciona la orden por índice, para acceder a sus propiedades.

```
bool  SelectByIndex(
   int  index   // índice del orden
   )
```

Valor devuelto

true - si se ejecuta correctamente, false - si la orden no se puede seleccionar.

© 2000-2025, MetaQuotes Ltd.

CHistoryOrderInfo

La clase CHistoryOrderInfo facilita el acceso al historial de propiedades de las órdenes.

Descripción

La clase CHistoryOrderInfo proporciona acceso al historial de propiedades de las órdenes.

Declaración

```
class CHistoryOrderInfo : public CObject
```

Título

```
#include <Trade\HistoryOrderInfo.mqh>
```

Jerarquía de herencia

CObject

CHistoryOrderInfo

Métodos de la clase por grupos

Acceso a las propiedades de tipo entero	
TimeSetup	Obtiene la hora de colocación de la orden
TimeSetupMsc	Recibe la hora de colocación de la orden en milisegundos, desde el 01-01-1970
OrderType	Obtiene el tipo de orden
OrderTypeDescription	Obtiene el tipo de orden en formato string
State	Obtiene el estado de la orden
StateDescription	Obtiene el estado de la orden en formato string
TimeExpiration	Obtiene la hora de expiración de la orden
TimeDone	Obtiene la hora de ejecución o cancelación de la orden
TimeDoneMsc	Recibe la hora de ejecución o cancelación de la orden en milisegundos desde el 01-01-1970
TypeFilling	Obtiene el tipo de ejecución de la orden
TypeFillingDescription	Obtiene el tipo de ejecución de la orden en formato string
TypeTime	Obtiene el tipo de orden en la hora de expiración
TypeTimeDescription	Obtiene el tipo de orden por hora de expiración en formato string
Magic	Obtiene el ID del experto que coloca la orden
PositionId	Obtiene el ID de la posición

© 2000-2025, MetaQuotes Ltd.

Acceso a las propiedades de tipo entero	
Acceso a las propiedades de tipo double	
VolumeInitial	Obtiene el volumen inicial de la orden
VolumeCurrent	Obtiene el volumen vacío de la orden
PriceOpen	Obtiene el precio de la orden
StopLoss	Obtiene el Stop Loss de la orden
TakeProfit	Obtiene el Take Profit de la orden
PriceCurrent	Obtiene el precio actual por el símbolo de la orden
PriceStopLimi	Obtiene el precio de establecimiento de la orden limit
Acceso a las propiedades de texto	
Symbol	Obtiene el símbolo de la orden
Comment	Obtiene el comentario de la orden
Acceso a las funciones del API de MQL5	
InfoInteger	Obtiene el valor de la propiedad de tipo entero especificada
InfoDouble	Obtiene el valor de la propiedad de tipo double especificada
InfoString	Obtiene el valor de la propiedad de tipo string especificada
Selección	
Ticket	Obtiene el ticket/selecciona la orden
SelectByIndex	Selecciona la orden por índice

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

TimeSetup

Obtiene la hora de colocación de la orden.

```
datetime  TimeSetup() const
```

Valor devuelto

Hora de colocación de la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeSetupMsc

Recibe la hora de colocación de la orden para su ejecución, en milisegundos desde el 01-01-1970.

```
ulong  TimeSetupMsc() const
```

Valor devuelto

Hora de colocación de la orden para su ejecución, en milisegundos desde el 01.01.1970.

Nota

La orden se tiene que seleccionar previamente con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

OrderType

Obtiene el tipo de la orden.

```
ENUM_ORDER_TYPE  OrderType() const
```

Valor devuelto

Tipo de orden (valor de la enumeración ENUM_ORDER_TYPE).

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeDescription

Obtiene el tipo de orden en formato string.

```
string  TypeDescription() const
```

Valor devuelto

Tipo de orden en formato string.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

State

Obtiene el estado de la orden.

```
ENUM_ORDER_STATE  State() const
```

Valor devuelto

Estado de la orden (valor de la enumeración ENUM_ORDER_STATE).

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

StateDescription

Obtiene el estado de la orden en formato string.

```
string  StateDescription() const
```

Valor devuelto

Estado de la orden en formato string.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeExpiration

Obtiene la hora de expiración de la orden.

```
datetime  TimeExpiration() const
```

Valor devuelto

Hora de expiración de la orden, establecida en la colocación.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeDone

Obtiene la hora de ejecución o cancelación.

```
datetime  TimeDone() const
```

Valor devuelto

Hora de ejecución o cancelación.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeDoneMsc

Recibe la hora de ejecución o cancelación de la orden en milisegundos desde el 01-01-1970.

```
ulong  TimeDoneMsc() const
```

Valor devuelto

Hora de ejecución o cancelación de la orden en milisegundos desde el 01-01-1970.

Nota

La orden se tiene que seleccionar previamente con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeFilling

Obtiene el tipo de ejecución de la orden.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Valor devuelto

Tipo de ejecución de la orden (valor de la enumeración ENUM_ORDER_TYPE_FILLING).

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeFillingDescription

Obtiene el tipo de ejecución de la orden en formato string.

```
string  TypeFillingDescription() const
```

Valor devuelto

Tipo de ejecución de la orden en formato string.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeTime

Obtiene el tipo de orden en la hora de expiración.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Valor devuelto

Tipo de orden en la hora de expiración (valor de la enumeración ENUM_ORDER_TYPE_TIME).

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeTimeDescription

Obtiene el tipo de orden por hora de expiración en formato string.

```
string  TypeTimeDescription() const
```

Valor devuelto

Tipo de orden por hora de expiración en formato string.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Magic

Obtiene el ID del Asesor Experto que coloca la orden.

```
long  Magic() const
```

Valor devuelto

ID del Asesor Experto que coloca la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PositionId

Obtiene el ID de la posición.

```
long  PositionId() const
```

Valor devuelto

ID de la posición correspondiente a la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

VolumeInitial

Obtiene el volumen inicial de la orden.

```
double  VolumeInitial() const
```

Valor devuelto

Obtiene el volumen inicial de la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

VolumeCurrent

Obtiene el volumen vacío de la orden.

```
double  VolumeCurrent() const
```

Valor devuelto

Volumen vacío de la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PriceOpen

Obtiene el precio de la orden.

```
double  PriceOpen() const
```

Valor devuelto

Precio de colocación de la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

StopLoss

Obtiene el precio Stop Loss de la orden.

```
double  StopLoss() const
```

Valor devuelto

Precio Stop Loss de la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TakeProfit

Obtiene el precio Take Profit de la orden.

```
double  TakeProfit() const
```

Valor devuelto

Precio Take Profit de la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PriceCurrent

Obtiene el precio actual del símbolo de la orden.

```
double  PriceCurrent() const
```

Valor devuelto

Precio actual del símbolo de la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PriceStopLimi

Obtiene el precio stop limit de la orden.

```
double  PriceStopLimit() const
```

Valor devuelto

Precio Stop Limit de la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Symbol

Obtiene el nombre del símbolo de la orden.

```
string  Symbol() const
```

Valor devuelto

Nombre del símbolo de la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Comment

Obtiene el comentario de la orden.

```
string  Comment() const
```

Valor devuelto

Comentario de la orden.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Obtiene el valor de la propiedad de tipo entero especificada.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,     // ID de la propiedad
   long&                        var          // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de tipo integer (valor de la enumeración ENUM_ORDER_PROPERTY_INTEGER).

var

[out] Referencia a la variable de tipo long, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Obtiene el valor de la propiedad de tipo double especificada.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,     // ID de la propiedad
   double&                     var         // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de tipo double (valor de la enumeración ENUM_ORDER_PROPERTY_DOUBLE).

var

[out] Referencia a la variable de tipo double para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoString

Obtiene el valor de la propiedad de tipo string especificada.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,     // ID de la propiedad
   string&                     var          // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de texto (valor de la enumeración ENUM_ORDER_PROPERTY_STRING).

var

[out] Referencia a la variable de tipo string, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La orden se tiene que seleccionar con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Ticket (Método Get)

Obtiene el ticket de la orden.

```
ulong  Ticket() const
```

Valor devuelto

Ticket de la orden.

Ticket (Método Set)

Selecciona la orden para poder trabajar con ella posteriormente.

```
void  Ticket(
   ulong  ticket      // ticket de la orden
   )
```

Parámetros

ticket

[in] Ticket de la orden.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

Selecciona la orden por índice, para acceder a sus propiedades.

```
bool  SelectByIndex(
   int  index   // índice del orden
   )
```

Valor devuelto

true - si se ejecuta correctamente, false - si la orden no se puede seleccionar.

© 2000-2025, MetaQuotes Ltd.

CPositionInfo

La clase CPositionInfo facilita el acceso a las propiedades de la posición abierta.

Descripción

La clase CPositionInfo proporciona acceso a las propiedades de la posición abierta.

Declaración

```
class CPositionInfo : public CObject
```

Título

```
#include <Trade\PositionInfo.mqh>
```

Jerarquía de herencia

CObject

CPositionInfo

Métodos de la clase por grupos

Acceso a las propiedades de tipo entero	
Time	Obtiene la hora de apertura de la posición
TimeMsc	Recibe la hora de apertura de la posición, en milisegundos desde el 01-01-1970
TimeUpdate	Recibe la hora de cambio de la posición, en segundos desde el 01-01-1970
TimeUpdateMsc	Recibe la hora de cambio de la posición, en milisegundos desde el 01-01-1970
PositionType	Obtiene el tipo de posición
TypeDescription	Obtiene el tipo de posición en formato string
Magic	Obtiene el ID del experto que abre la posición
Identificador	Obtiene el ID de la posición
Acceso a las propiedades de tipo double	
Volume	Obtiene el volumen de la posición
PriceOpen	Obtiene el precio de apertura de la posición
StopLoss	Obtiene el precio de Stop Loss de la posición
TakeProfit	Obtiene el precio de Take Profit de la posición

© 2000-2025, MetaQuotes Ltd.

Acceso a las propiedades de tipo entero	
PriceCurrent	Obtiene el precio actual por símbolo de la posición
Commission	Obtiene la cantidad de comisión por posición
Swap	Obtiene la cantidad de swap por posición
Profit	Obtiene el beneficio actual por posición
Acceso a las propiedades de texto	
Symbol	Obtiene el nombre del símbolo de la posición
Comment	Obtiene el comentario de la posición
Acceso a las funciones del API de MQL5	
InfoInteger	Obtiene el valor de la propiedad de tipo entero especificada
InfoDouble	Obtiene el valor de la propiedad de tipo double especificada
InfoString	Obtiene el valor de la propiedad de tipo string especificada
Selección	
Select	Selecciona la posición
SelectByIndex	Selecciona la posición por índice
SelectByMagic	Selects a position with the specified symbol name and magic number
SelectByTicket	Selects the position by ticket
State	
StoreState	Guarda los parámetros de la posición
CheckState	Comprueba los parámetros actuales con los parámetros guardados

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Time

Obtiene la hora de apertura de la posición.

```
datetime  Time() const
```

Valor devuelto

Hora de apertura de la posición.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeMsc

Recibe la hora de apertura de la posición en milisegundos desde el 01-01-1970.

```
ulong  TimeMsc() const
```

Valor devuelto

Hora de apertura de la posición en milisegundos desde el 01-01-1970.

Nota

La posición se tiene que seleccionar previamente con los métodos Select (por símbolo) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeUpdate

Recibe la hora de cambio de la posición en segundos desde el 01-01-1970.

```
datetime  TimeUpdate() const
```

Valor devuelto

Hora de cambio de la posición en segundos desde el 01-01-1970.

Nota

La posición se tiene que seleccionar previamente con los métodos Select (por símbolo) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeUpdateMsc

Recibe la hora de cambio de la posición, en milisegundos desde el 01-01-1970.

```
ulong  TimeUpdateMsc() const
```

Valor devuelto

Hora de cambio de la posición, en milisegundos desde el 01-01-1970.

Nota

La posición se tiene que seleccionar previamente con los métodos Select (por símbolo) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PositionType

Obtiene el tipo de la posición.

```
ENUM_POSITION_TYPE  PositionType() const
```

Valor devuelto

Tipo de posición (valor de la enumeración ENUM_POSITION_TYPE).

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeDescription

Obtiene el tipo de la posición en formato string.

```
string  TypeDescription() const
```

Valor devuelto

Tipo de la posición en formato string.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Magic

Obtiene el ID del Asesor Experto que abre la posición.

```
long  Magic() const
```

Valor devuelto

ID del Asesor Experto que abre la posición.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Identificador

Obtiene el ID de la posición.

```
long  Identifier() const
```

Valor devuelto

ID de la posición.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Volume

Obtiene el volumen de la posición.

```
double  Volume() const
```

Valor devuelto

Volumen de la posición.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PriceOpen

Obtiene el precio de apertura de la posición.

```
double  PriceOpen() const
```

Valor devuelto

Precio de apertura de la posición.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

StopLoss

Obtiene el precio de Stop Loss de la posición.

```
double  StopLoss() const
```

Valor devuelto

Precio de Stop Loss de la posición.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TakeProfit

Obtiene el precio de Take Profit de la posición.

```
double  TakeProfit() const
```

Valor devuelto

Precio de Take Profit de la posición.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PriceCurrent

Obtiene el precio actual por el símbolo de la posición.

```
double  PriceCurrent() const
```

Valor devuelto

Precio actual por el símbolo de la posición.

© 2000-2025, MetaQuotes Ltd.

Commission

Obtiene la cantidad de comisión de la posición.

```
double  Commission() const
```

Valor devuelto

Cantidad de comisión de la posición (en la divisa del depósito).

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Swap

Obtiene la cantidad de swap de la posición.

```
double  Swap() const
```

Valor devuelto

Cantidad de swap de la posición (en la divisa del depósito).

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Profit

Obtiene el beneficio actual de la posición.

```
double  Profit() const
```

Valor devuelto

Beneficio actual de la posición (en la divisa del depósito).

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Symbol

Obtiene el nombre del símbolo de la posición.

```
string  Symbol() const
```

Valor devuelto

Nombre del símbolo de la posición.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Comment

Obtiene el comentario de la posición.

```
string  Comment() const
```

Valor devuelto

Comentario de la posición.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Obtiene el valor de la propiedad de tipo entero especificada.

```
bool  InfoInteger(
   ENUM_POSITION_PROPERTY_INTEGER  prop_id,     // ID de la propiedad
   long&                           var          // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de tipo integer (valor de la enumeración ENUM_POSITION_PROPERTY_INTEGER).

var

[out] Referencia a la variable de tipo long, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Obtiene el valor de la propiedad de tipo double especificada.

```
bool  InfoDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  prop_id,      // ID de la propiedad
   double&                        var          // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de tipo double (valor de la enumeración ENUM_POSITION_PROPERTY_DOUBLE).

var

[in] Referencia a la variable de tipo double, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoString

Obtiene el valor de la propiedad de tipo string especificada.

```
bool  InfoString(
   ENUM_POSITION_PROPERTY_STRING  prop_id,      // ID de la propiedad
   string&                        var           // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de texto (valor de la enumeración ENUM_POSITION_PROPERTY_STRING).

var

[out] Referencia a la variable de tipo string, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La posición se tiene que seleccionar con los métodos Select (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Select

Selecciona la posición, para poder trabajar con ella más adelante.

```
bool  Select(
   const string  symbol      // símbolo
   )
```

Parámetros

symbol

[in] Símbolo de la posición.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

Selecciona la posición por índice, para poder acceder a sus propiedades posteriormente.

```
bool  SelectByIndex(
   int  index   // índice de la posición
   );
```

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede seleccionar la posición.

© 2000-2025, MetaQuotes Ltd.

SelectByMagic

Elige una posición según el nombre del instrumento y el número mágico para trabajar posteriormente con ella.

```
bool  SelectByMagic(
   const string  symbol, // nombre del instrumento
   const ulong   magic   // número mágico
   );
```

Parámetros

symbol

[in] Nombre del instrumento.

magic

[in] Número mágico de la posición.

Valor devuelto

Devuelve true en caso de éxito, o false, si no se ha elegido una posición.

© 2000-2025, MetaQuotes Ltd.

SelectByTicket

Elige una posición según el ticket para trabajar posteriormente con ella.

```
bool  SelectByTicket(
   ulong  ticket      // ticket de la posición
   )
```

Parámetros

ticket

[in] Ticket de la posición.

Valor devuelto

Devuelve true en caso de éxito, o false, si no se ha elegido una posición.

© 2000-2025, MetaQuotes Ltd.

StoreState

Guarda los parámetros de la posición.

```
void  StoreState()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

CheckState

Comprueba los parámetros actuales con los parámetros guardados.

```
bool  CheckState()
```

Valor devuelto

true - si los parámetros de la posición cambian desde la última llamada al método StoreState() , en caso contrario - false.

© 2000-2025, MetaQuotes Ltd.

CDealInfo

La clase CDealInfo facilita el acceso a las propiedades de la transacción.

Descripción

La clase CDealInfo proporciona acceso a las propiedades de la transacción.

Declaración

```
class CDealInfo : public CObject
```

Título

```
#include <Trade\DealInfo.mqh>
```

Jerarquía de herencia

CObject

CDealInfo

Métodos de la clase por grupos

Acceso a las propiedades de tipo entero	
Order	Obtiene la orden por la que se ejecuta la transacción
Time	Obtiene la hora de ejecución de la transacción
TimeMsc	Recibe la hora de ejecución de la transacción en milisegundos, desde el 01.01.1970
DealType	Obtiene el tipo de transacción
TypeDescription	Obtiene el tipo de transacción en formato string
Entry	Obtiene la dirección de la transacción
EntryDescription	Obtiene la dirección de la transacción en formato string
Magic	Obtiene el ID del experto que ejecuta la transacción
PositionId	Obtiene el ID de la posición correspondiente a la transacción
Acceso a las propiedades de tipo double	
Volume	Obtiene el volumen de la transacción
Price	Obtiene el precio de la transacción
Commision	Obtiene la comisión por la transacción
Swap	Obtiene la cantidad de swap cuando se cierra la posición
Profit	Obtiene el resultado financiero de la transacción

© 2000-2025, MetaQuotes Ltd.

Acceso a las propiedades de tipo entero	
Acceso a las propiedades de texto	
Symbol	Obtiene el nombre del símbolo de la transacción
Comment	Obtiene el comentario de la transacción
Acceso a las funciones del API de MQL5	
InfoInteger	Obtiene el valor de la propiedad de tipo entero especificada
InfoDouble	Obtiene el valor de la propiedad de tipo double especificada
InfoString	Obtiene el valor de la propiedad de tipo string especificada
Selección	
Ticket	Obtiene el ticket/selecciona la transacción
SelectByIndex	Selecciona la transacción por índice

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Order

Obtiene la orden por la que se ejecuta la transacción.

```
long  Order() const
```

Valor devuelto

Orden por la que se ejecuta la transacción.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Time

Obtiene la hora de ejecución de la transacción.

```
datetime  Time() const
```

Valor devuelto

Hora de ejecución de la transacción.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TimeMsc

Recibe la hora de ejecución de la transacción en milisegundos, desde el 01-01-1970.

```
ulong  TimeMsc() const
```

Valor devuelto

La hora de ejecución de la transacción en milisegundos desde el 01-01-1970.

Nota

La transacción se tiene que seleccionar previamente con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

DealType

Obtiene el tipo de la transacción.

```
ENUM_DEAL_TYPE  DealType() const
```

Valor devuelto

Tipo de transacción (valor de la enumeración ENUM_DEAL_TYPE).

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

TypeDescription

Obtiene el tipo de transacción en formato string.

```
string  TypeDescription() const
```

Valor devuelto

Tipo de transacción en formato string.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Entry

Obtiene la dirección de la transacción.

```
ENUM_DEAL_ENTRY  Entry() const
```

Valor devuelto

Dirección de la transacción (valor de la enumeración ENUM_DEAL_ENTRY).

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

EntryDescription

Obtiene la dirección de la transacción en formato cadena.

```
string  EntryDescription() const
```

Valor devuelto

Dirección de la transacción en formato cadena.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Magic

Obtiene el identificador del Asesor Experto que ejecuta la señal.

```
long  Magic() const
```

Valor devuelto

ID del Asesor Experto que ejecuta la señal.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

PositionId

Obtiene el identificador de la posición correspondiente a la transacción.

```
long  PositionId() const
```

Valor devuelto

ID de la posición correspondiente a la transacción.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Volume

Obtiene el volumen de la transacción.

```
double  Volume() const
```

Valor devuelto

Volumen de la transacción.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Price

Obtiene el precio de la transacción.

```
double  Price() const
```

Valor devuelto

Precio de la transacción.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Commission

Obtiene la comisión de la transacción.

```
double  Commission() const
```

Valor devuelto

Comisión de la transacción.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Swap

Obtiene el swap cuando se cierra la posición.

```
double  Swap() const
```

Valor devuelto

Cantidad de swap cuando se cierra la posición.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Profit

Obtiene el resultado financiero de la transacción.

```
double  Profit() const
```

Valor devuelto

Resultado financiero de la transacción (en la divisa del depósito).

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Symbol

Obtiene el nombre del símbolo de la transacción.

```
string  Symbol() const
```

Valor devuelto

Nombre del símbolo de la transacción.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Comment

Obtiene el comentario de la transacción.

```
string  Comment() const
```

Valor devuelto

Comentario de la transacción.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Obtiene el valor de la propiedad de tipo entero especificada.

```
bool  InfoInteger(
   ENUM_DEAL_PROPERTY_INTEGER  prop_id,      // ID de la propiedad
   long&                       var           // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de tipo integer (valor de la enumeración ENUM_DEAL_PROPERTY_INTEGER).

var

[out] Referencia a la variable de tipo long, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Obtiene el valor de la propiedad de tipo double especificada.

```
bool  InfoDouble(
   ENUM_DEAL_PROPERTY_DOUBLE  prop_id,      // ID de la propiedad
   double&                    var           // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de tipo double (valor de la enumeración ENUM_DEAL_PROPERTY_DOUBLE).

var

[in] Referencia a la variable de tipo double, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

InfoString

Obtiene el valor de la propiedad de tipo string especificada.

```
bool  InfoString(
   ENUM_DEAL_PROPERTY_STRING  prop_id,      // ID de la propiedad
   string&                    var           // referencia a la variable
   ) const
```

Parámetros

prop_id

[in] ID de la propiedad de texto (valor de la enumeración ENUM_DEAL_PROPERTY_STRING).

var

[out] Referencia a la variable de tipo string, para poner el resultado.

Valor devuelto

true - si se ejecuta correctamente, false - si no se puede obtener el valor de la propiedad.

Nota

La transacción tiene que seleccionarse con los métodos Ticket (por ticket) o SelectByIndex (por índice).

© 2000-2025, MetaQuotes Ltd.

Ticket (Método Get)

Obtiene el ticket de la transacción.

```
ulong  Ticket() const
```

Valor devuelto

Ticket de la transacción.

Ticket (Método Set)

Selecciona la posición, para poder trabajar con ella más adelante.

```
void  Ticket(
   ulong  ticket      // ticket
   )
```

Parámetros

ticket

[in] Ticket de la transacción.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

Selecciona la transacción por su índice, para poder acceder a sus propiedades.

```
bool  SelectByIndex(
   int  index   // índice del orden
   )
```

Valor devuelto

true - si se ejecuta correctamente, false - si la transacción no se puede seleccionar.

© 2000-2025, MetaQuotes Ltd.

CTrade

La clase CTrade facilita el acceso a las funciones de trading.

Descripción

La clase CTrade proporciona acceso a las funciones de trading.

Declaración

```
class CTrade : public CObject
```

Título

```
#include <Trade\Trade.mqh>
```

Jerarquía de herencia

CObject

CTrade

Descendientes directos

CExpertTrade

Métodos de la clase por grupos

Parámetros de configuración	
LogLevel	Establece el nivel de logging
SetExpertMagicNumber	Establece el identificador del experto
SetDeviationInPoints	Establece la desviación permitida
SetTypeFilling	Establece el tipo de relleno de la orden
SetTypeFillingBySymbol	Establece el tipo de orden según la ejecución, de acuerdo con los ajustes del símbolo indicado
SetAsyncMode	Establece el modo asíncrono para las operaciones de trading
SetMarginMode	Establece el modo de cálculo del margen de acuerdo con los ajustes de la cuenta actual
Operaciones con órdenes	
OrderOpen	Coloca la orden pendiente con los parámetros establecidos
OrderModify	Modifica los parámetros de la orden pendiente
OrderDelete	Borra la orden pendiente
Operaciones con las posiciones	

© 2000-2025, MetaQuotes Ltd.

Parámetros de configuración	
PositionOpen	Abre la posición con los parámetros establecidos
PositionModify	Modifica los parámetros de la posición
PositionClose	Cierra la posición
PositionClosePartial	Cierra parcialmente una posición del símbolo indicado o con un ticket específico.
PositionCloseBy	Cierra una posición con la posición opuesta según el ticket indicado
Métodos adicionales	
Buy	Abre la posición larga con los parámetros especificados
Sell	Abre la posición corta con los parámetros especificados
BuyLimit	Coloca la orden pendiente de tipo Buy Limit con los parámetros especificados
BuyStop	Coloca la orden pendiente de tipo Buy Stop con los parámetros especificados
SellLimit	Coloca la orden pendiente de tipo Sell Limit con los parámetros especificados
SellStop	Coloca la orden pendiente de tipo Sell Stop con los parámetros especificados
Acceso a los parámetros de la última solicitud	
Request	Obtiene la copia de la estructura de la última solicitud
RequestAction	Obtiene el tipo de operación de trading
RequestActionDescription	Obtiene el tipo de operación de trading en formato string
RequestMagic	Obtiene el número mágico del Asesor Experto
RequestOrder	Obtiene el ticket de la orden de la última petición
RequestSymbol	Obtiene el nombre del símbolo de la última petición
RequestVolume	Obtiene el volumen de trading (en lotes) de la última petición
RequestPrice	Obtiene el precio de la última petición
RequestStopLimit	Obtiene el precio de la orden pendiente de tipo Stop Limit de la última petición
RequestSL	Obtiene el precio de Stop Loss de la orden de la última petición
RequestTP	Obtiene el precio de Take Profit de la orden de la última petición
RequestDeviation	Obtiene la desviación del precio de la orden de la última petición

© 2000-2025, MetaQuotes Ltd.

Parámetros de configuración	
RequestType	Obtiene el tipo de orden de la última petición
RequestTypeDescription	Obtiene el tipo de orden, en formato string, de la última petición
RequestTypeFilling	Obtiene el tipo de relleno de la orden de la última petición
RequestTypeFillingDescription	Obtiene el tipo de relleno de la orden, en formato cadena, de la última petición
RequestTypeTime	Obtiene el período de validez de la orden de la última petición
RequestTypeTimeDescription	Obtiene el período de validez de la orden (como string) de la última petición
RequestExpiration	Obtiene la hora de expiración de la orden de la última petición
RequestComment	Obtiene el comentario de la orden de la última petición
RequestPosition	Obtiene el ticket de la posición
RequestPositionBy	Obtiene el ticket de la posición opuesta
Acceso a la última solicitud de comprobación de resultados	
CheckResult	Obtiene la copia de la estructura de la última solicitud de comprobación del resultado.
CheckResultRetcode	Obtiene el valor del campo retcode de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta
CheckResultRetcodeDescription	Obtiene la descripción string del campo retcode de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta
CheckResultBalance	Obtiene el valor del campo balance de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta
CheckResultEquity	Obtiene el valor del campo equity de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta
CheckResultProfit	Obtiene el valor del campo profit de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta
CheckResultMargin	Obtiene el valor del campo margin de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta
CheckResultMarginFree	Obtiene el valor del campo margin_free de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta
CheckResultMarginLevel	Obtiene el valor del campo margin_level de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta

© 2000-2025, MetaQuotes Ltd.

Parámetros de configuración	
CheckResultComment	Obtiene el valor del campo comment de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta
Accede a los últimos resultados de la ejecución de la petición	
Result	Obtiene la copia de la estructura del último resultado de la solicitud
ResultRetcode	Obtiene el código del resultado de la solicitud
ResultRetcodeDescription	Obtiene el código del resultado de la solicitud, en formato texto
ResultDeal	Obtiene el ticket de la transacción
ResultOrder	Obtiene el ticket de la orden
ResultVolume	Obtiene el volumen de la transacción o de la orden
ResultPrice	Obtiene el precio confirmado por el broker
ResultBid	Obtiene el precio bid actual (la recotización)
ResultAsk	Obtiene el precio ask actual (la recotización)
ResultComment	Obtiene el comentario del broker
Métodos auxiliares	
PrintRequest	Imprime los últimos parámetros de la petición
PrintResult	Imprime los resultados de la última petición
FormatRequest	Prepara la cadena con formato, con los parámetros de la última petición
FormatRequestResult	Prepara la cadena con formato, con los resultados de ejecución de la última solicitud

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

LogLevel

Establece el nivel de logging de los mensajes.

```
void  LogLevel(
   ENUM_LOG_LEVELS  log_level      // nivel de logging
   )
```

Parámetros

log_level

[in] Nivel de logging.

Valor devuelto

Ninguno.

Nota

LOG_LEVEL_NO e inferior, desactiva la visualización de los mensajes (establecido automáticamente en el modo de optimización). LOG_LEVEL_ERRORS solamente imprime los mensajes de error (valor predeterminado). LOG_LEVEL_ALL y superior, activa la impresión de todos los mensajes (establecido automáticamente en el modo de test).

ENUM_LOG_LEVELS

Identificador	Descripción	Valor
LOG_LEVEL_NO	No se muestran los mensajes	0
LOG_LEVEL_ERRORS	Solo se muestran los mensajes de error	1
LOG_LEVEL_ALL	Se muestran todos los mensajes	2

© 2000-2025, MetaQuotes Ltd.

SetExpertMagicNumber

Establece el ID del experto.

```
void  SetExpertMagicNumber(
   ulong  magic      // identificador
   )
```

Parámetros

magic

[in] Nuevo ID del experto.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

SetDeviationInPoints

Establece la desviación permitida.

```
void  SetDeviationInPoints(
   ulong  deviation      // desviación
   )
```

Parámetros

deviation

[in] Desviación permitida.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

SetTypeFilling

Establece el tipo de relleno de la orden.

```
void  SetTypeFilling(
   ENUM_ORDER_TYPE_FILLING  filling      // tipo de relleno de la orden
   )
```

Parámetros

filling

[in] Tipo de relleno de la orden (valor de la enumeración ENUM_ORDER_TYPE_FILLING).

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

SetTypeFillingBySymbol

Define el tipo de orden según la ejecución, de acuerdo con los ajustes del símbolo indicado.

```
bool  SetTypeFillingBySymbol(
   const string   symbol      // nombre del instrumento financiero
   )
```

Parámetros

symbol

[in] Nombre del instrumento en el que SYMBOL_FILLING_MODE contiene las políticas permitidas de ejecución de órdenes.

Valor devuelto

true - en caso de que la ejecución tenga éxito, false - si no se ha logrado definir la política de ejecución.

Nota

Si para el símbolo están permitidas simultáneamente las políticas SYMBOL_FILLING_FOK y SYMBOL_FILLING_IOC, para la orden se establecerá el valor ORDER_FILLING_FOK.

© 2000-2025, MetaQuotes Ltd.

SetAsyncMode

Establece el modo asíncrono para las operaciones de trading.

```
void  SetAsyncMode(
   bool  mode      // bandera de modo asíncrono
   )
```

Parámetros

mode

[in] Bandera de modo asíncrono.

Valor devuelto

Ninguno.

Nota

Este modo se utiliza en las operaciones de trading asíncronas, donde la petición enviada no espera la respuesta del servidor (ver OrderSendAsync).

© 2000-2025, MetaQuotes Ltd.

SetMarginMode

Define el modo de cálculo del margen de acuerdo con los ajustes de la cuenta actual.

```
void  SetMarginMode()
```

Valor devuelto

Ninguno.

Observación

El modo de cálculo del margen se indica en ENUM_ACCOUNT_MARGIN_MODE.

© 2000-2025, MetaQuotes Ltd.

OrderOpen

Coloca la orden pendiente con los parámetros establecidos.

```
bool  OrderOpen(
   const string          symbol,          // símbolo
   ENUM_ORDER_TYPE       order_type,      // tipo de orden
   double                volume,          // volumen de la orden
   double                limit_price,     // precio StopLimit
   double                price,           // precio de ejecución
   double                sl,              // precio Stop Loss
   double                tp,              // precio Take Profit
   ENUM_ORDER_TYPE_TIME  type_time,       // tipo por expiración
   datetime              expiration,      // expiración
   const string          comment=""       // comentario
   )
```

Parámetros

symbol

[in] Nombre del instrumento de trading.

order_type

[in] Tipo de orden (valor de la enumeración ENUM_ORDER_TYPE).

volume

[in] Volumen de la orden solicitado.

limit_price

[in] Precio por el que la orden StopLimit se colocará.

price

[in] Precio por el que la orden se debe ejecutar.

sl

[in] Precio al que el Stop Loss se disparará.

tp

[in] Precio al que el Take Profit se disparará.

type_time

[in] Tipo de orden por ejecución (valor de la enumeración ENUM_ORDER_TYPE_TIME).

expiration

[in] Fecha de expiración de la orden pendiente.

comment=""

[in] Comentario de la orden.

Valor devuelto

true - en caso de que la comprobación de las estructuras básicas sea exitosa; en caso contrario - false.

© 2000-2025, MetaQuotes Ltd.

Nota

Si se completa correctamente el método OrderSend(...), esto no implica siempre una ejecución exitosa de la operación de trading. Es necesario comprobar el resultado de la petición de trading (código de retorno del servidor) con ResultRetcode(), y el valor devuelto por ResultOrder().

© 2000-2025, MetaQuotes Ltd.

OrderModify

Modifica los parámetros de la orden pendiente.

```
bool  OrderModify(
   ulong                ticket,         // ticket de la orden
   double               price,          // precio de ejecución
   double               sl,             // precio Stop Loss
   double               tp,             // precio Take Profit
   ENUM_ORDER_TYPE_TIME type_time,      // tipo por expiración
   datetime             expiration,     // expiración
   double               stoplimit       // precio de la orden Limit
   )
```

Parámetros

ticket

[in] Ticket de la orden.

price

[in] El precio nuevo por el que la orden se tiene que ejecutar (o el valor anterior, si el cambio no es necesario).

sl

[in] El precio nuevo por el que el Stop Loss se disparará (o el valor anterior, si el cambio no es necesario).

tp

[in] El precio nuevo por el que el Take Profit se disparará (o el valor anterior, si el cambio no es necesario).

type_time

[in] El nuevo tipo de orden por expiración (o el valor anterior, si el cambio no es necesario), valor de la enumeración ENUM_ORDER_TYPE_TIME.

expiration

[in] La nueva fecha de expiración de la orden pendiente (o el valor anterior, si el cambio no es necesario).

stoplimit

[in] Nuevo precio utilizado para establecer una orden Limit cuando el precio alcanza el valor del precio. Especificado solo por órdenes StopLimit.

Valor devuelto

true - en caso de que la comprobación de las estructuras básicas sea exitosa; en caso contrario - false.

Nota

Si se completa correctamente el método OrderModify(...), esto no implica siempre una ejecución exitosa de la operación de trading. Es necesario comprobar el resultado de la petición de trading (código de retorno del servidor) con ResultRetcode().

© 2000-2025, MetaQuotes Ltd.

OrderDelete

Borra la orden pendiente.

```
bool  OrderDelete(
   ulong  ticket      // ticket de la orden
   )
```

Parámetros

ticket

[in] Ticket de la orden.

Valor devuelto

true - en caso de que la comprobación de las estructuras básicas sea exitosa; en caso contrario - false.

Nota

Si se completa correctamente el método OrderDelete(...), esto no implica siempre una ejecución exitosa de la operación de trading. Es necesario comprobar el resultado de la petición de trading (código de retorno del servidor) con ResultRetcode().

© 2000-2025, MetaQuotes Ltd.

PositionOpen

Abre la posición con los parámetros establecidos.

```
bool  PositionOpen(
   const string     symbol,          // símbolo
   ENUM_ORDER_TYPE  order_type,      // tipo de orden para abrir la posición
   double           volume,          // volumen de la posición
   double           price,           // precio de ejecución
   double           sl,              // precio Stop Loss
   double           tp,              // precio Take Profit
   const string     comment=""       // comentario
   )
```

Parámetros

symbol

[in] Nombre del instrumento de trading por el que se quiere abrir la posición.

order_type

[in] Tipo de orden (operación de trading) para abrir la posición (valor de la enumeración ENUM_ORDER_TYPE).

volume

[in] Volumen de la posición solicitada.

price

[in] Precio al que la posición se tiene que abrir.

sl

[in] Precio al que el Stop Loss se disparará.

tp

[in] Precio al que el Take Profit se disparará.

comment=""

[in] Comentario de la posición.

Valor devuelto

true - en caso de que la comprobación de las estructuras básicas sea exitosa; en caso contrario - false.

Nota

Si se completa correctamente el método PositionOpen(...), esto no implica siempre una ejecución exitosa de la operación de trading. Es necesario comprobar el resultado de la petición de trading (código de retorno del servidor) con ResultRetcode() y el valor devuelto por ResultDeal().

© 2000-2025, MetaQuotes Ltd.

PositionModify

Modifica los parámetros de la posición por el símbolo especificado.

```
bool  PositionModify(
   const string  symbol,      // símbolo
   double        sl,          // precio Stop Loss
   double        tp           // precio Take Profit
   )
```

Cambia los parámetros de una posición según el ticket indicado.

```
bool  PositionModify(
   const ulong   ticket,      // position ticket
   double        sl,          // Stop Loss price
   double        tp           // Take Profit price
   )
```

Parámetros

symbol

[in] Nombre del instrumento por el que se quiere modificar la posición.

ticket

[in] Ticket de la posición que se presupone se va a modificar.

sl

[in] El precio nuevo por el que el Stop Loss se disparará (o el valor anterior, si el cambio no es necesario).

tp

[in] El precio nuevo por el que el Take Profit se disparará (o el valor anterior, si el cambio no es necesario).

Valor devuelto

true - en caso de que la comprobación de las estructuras básicas sea exitosa; en caso contrario - false.

Nota

El hecho de que el trabajo del método PositionModify(...) haya finalizado, no siempre significa que la operación comercial haya sido ejecutada con éxito. Es necesario comprobar el resultado de la ejecución de la solicitud comercial (código de retorno del servidor comercial) llamando al método ResultRetcode().

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

© 2000-2025, MetaQuotes Ltd.

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones. En este caso, PositionModify cambiará la posición con el ticket menor.

© 2000-2025, MetaQuotes Ltd.

PositionClose

Cierra la posición por el símbolo especificado.

```
bool  PositionClose(
   const string  symbol,                  // símbolo
   ulong         deviation=ULONG_MAX      // desviación
   )
```

Closes a position with the specified ticket.

```
bool  PositionClose(
   const ulong   ticket,                  // Position ticket
   ulong         deviation=ULONG_MAX      // Deviation
   )
```

Parámetros

symbol

[in] Nombre del instrumento por el que se quiere cerrar la posición.

ticket

[in] Ticket de la posición cerrada.

deviation=ULONG_MAX

[in] Desviación máxima del precio actual (en puntos).

Valor devuelto

true - en caso de que la comprobación de las estructuras básicas sea exitosa; en caso contrario - false.

Nota

El hecho de que el trabajo del método PositionClose(...) haya finalizado, no siempre significa que la operación comercial haya sido ejecutada con éxito. Es necesario comprobar el resultado de la ejecución de la solicitud comercial (código de retorno del servidor comercial) llamando al método ResultRetcode().

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), de cada símbolo pueden abrirse a la vez varias posiciones. En este caso, PositionClose cerrará la posición con el ticket menor.

© 2000-2025, MetaQuotes Ltd.

PositionClosePartial

Cierra parte de la posición del símbolo indicado en el registro de "cobertura".

```
bool  PositionClosePartial(
   const string  symbol,                  // símbolo
   const double  volume,                  // volumen
   ulong         deviation=ULONG_MAX      // desviación
   )
```

Cierra parte de la posición con ticket indicado en el registro de "cobertura".

```
bool  PositionClosePartial(
   const ulong   ticket,                  // ticket de la posición
   const double  volume,                  // volumen
   ulong         deviation=ULONG_MAX      // desviación
   )
```

Parámetros

symbol

[in] Nombre del instrumento comercial del que se supone cerrar parte de la posición. Si para el cierre parcial de posición se indica el símbolo (no el ticket), se elegirá la primera posición encontrada de dicho símbolo que tenga el MagicNumber indicado (identificador del experto). Por eso, en ocasiones es mejor usar la variante PositionClosePartial() indicando el ticket de la posición.

volume

[in] Volumen en el que hay que reducir la posición. Si el valor es superior al volumen de la posición parcialmente cerrada, la posición se cerrará por completo. No se abrirá una nueva posición en dirección opuesta.

ticket

[in] Ticket de la posición cerrada.

deviation=ULONG_MAX

[in] Desviación máxima a partir del precio actual (en puntos).

Valor devuelto

true - si la comprobación básica de las estructuras ha tenido éxito, de lo contrario, false.

Nota

El hecho de que el trabajo del método PositionClose(...) haya finalizado, no siempre significa que la operación comercial haya sido ejecutada con éxito. Es necesario comprobar el resultado de la ejecución de la solicitud comercial (código de retorno del servidor comercial) llamando al método ResultRetcode().

En el registro de posiciones con "compensación" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING y ACCOUNT_MARGIN_MODE_EXCHANGE) de cada símbolo en cualquier momento solo puede abrirse una posición, que es el resultado de una o más operaciones. Es mejor no confundir las posiciones

© 2000-2025, MetaQuotes Ltd.

con las órdenes pendientes en activo, que también se muestran en la pestaña "Trading" en el panel "Herramientas".

En el caso de que las posiciones se representen independientemente (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) de cada símbolo pueden abrirse a la vez varias posiciones. En este caso, PositionClose cerrará la posición con el ticket menor.

© 2000-2025, MetaQuotes Ltd.

PositionCloseBy

Cierra la posición con el ticket indicado de la posición en dirección opuesta

```
bool  PositionCloseBy(
   const ulong   ticket,        // Ticket de la posición
   const ulong   ticket_by      // Ticket de la posición opuesta
   )
```

Parámetros

ticket

[in] Ticket de la posición a cerrar.

ticket_by

[in] Ticket de la posición opuesta, que se usa para el cierre.

Valor retornado

true - en el caso de que la comprobación básica de las estructuras haya tenido éxito, de lo contrario, false.

Advertencia

El hecho de que el método PositionCloseBy(...) haya finalizado con éxito su trabajo no significa siempre que la operación comercial se haya realizado con éxito. Es necesario comprobar el resultado de la ejecución de la solicitud comercial (código de retorno del servidor comercial) llamando al método ResultRetcode().

© 2000-2025, MetaQuotes Ltd.

Buy

Abre la posición larga con los parámetros especificados.

```
bool  Buy(
   double         volume,          // volumen de la posición
   const string   symbol=NULL,     // símbolo
   double         price=0.0,       // precio
   double         sl=0.0,          // precio stop loss
   double         tp=0.0,          // precio take profit
   const string   comment=""       // comentario
   )
```

Parámetros

volume

[in] Volumen de la posición.

symbol=NULL

[in] Símbolo de la posición. Si el símbolo no se especifica, se utiliza el símbolo actual.

price=0.0

[in] Precio. Si el precio no se especifica, se utiliza el precio Ask actual.

sl=0.0

[in] Precio Stop Loss.

tp=0.0

[in] Precio Take Profit.

comment=""

[in] Comentario.

Valor devuelto

true - si las estructuras se pueden comprobar correctamente; en caso contrario - false.

Nota

Si se completa correctamente el método Buy(...), esto no siempre implica una ejecución exitosa de la operación de trading. Es necesario comprobar el resultado de la petición de trading (el código de retorno del servidor) con el valor de ResultRetcode(), y el valor devuelto por ResultDeal().

© 2000-2025, MetaQuotes Ltd.

Sell

Abre la posición corta con los parámetros especificados.

```
bool  Sell(
   double         volume,           // volumen de la posición
   const string   symbol=NULL,      // símbolo
   double         price=0.0,        // precio
   double         sl=0.0,           // precio stop loss
   double         tp=0.0,           // precio take profit
   const string   comment=""        // comentario
   )
```

Parámetros

volume

[in] Volumen de la posición.

symbol=NULL

[in] Símbolo de la posición. Si el símbolo no se especifica, se utiliza el símbolo actual.

price=0.0

[in] Precio. Si el precio no se especifica, se utiliza el precio Bid actual del mercado.

sl=0.0

[in] Precio Stop Loss.

tp=0.0

[in] Precio Take Profit.

comment=""

[in] Comentario.

Valor devuelto

true - si las estructuras se pueden comprobar correctamente; en caso contrario - false.

Nota

Si se completa correctamente el método Sell(...), esto no siempre implica una ejecución exitosa de la operación de trading. Es necesario comprobar el resultado de la petición de trading (el código de retorno del servidor) con el valor de ResultRetcode(), y el valor devuelto por ResultDeal().

© 2000-2025, MetaQuotes Ltd.

BuyLimit

Coloca la orden pendiente de tipo Buy Limit (comprar en el precio, más bajo que el precio actual del mercado) con los parámetros especificados.

```
bool  BuyLimit(
   double                volume,                    // volumen de la orden
   double                price,                     // precio de la orden
   const string          symbol=NULL,               // símbolo
   double                sl=0.0,                    // precio stop loss
   double                tp=0.0,                    // precio take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // vida de la orden
   datetime              expiration=0,              // hora de expiración de la orc
   const string          comment=""                 // comentario
   )
```

Parámetros

volume

[in] Volumen de la orden.

price

[in] Precio de la orden.

symbol=NULL

[in] Símbolo de la orden. Si el símbolo no se especifica, se utiliza el símbolo actual.

sl=0.0

[in] Precio Stop Loss.

tp=0.0

[in] Precio Take Profit.

type_time=ORDER_TIME_GTC

[in] Vida de la orden (valor de la enumeración ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Hora de expiración de la orden (se utiliza solo si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Comentario de la orden.

Valor devuelto

true - si las estructuras se pueden comprobar correctamente; en caso contrario - false.

Nota

Si se completa correctamente el método BuyLimit(...), esto no implica siempre una ejecución exitosa de la operación de trading. Es necesario comprobar el resultado de la petición de trading (el código de retorno del servidor) con el valor de ResultRetcode(), y el valor devuelto por ResultOrder().

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

BuyStop

Coloca la orden pendiente de tipo Buy Stop (comprar en el precio, más alto que el precio actual del mercado) con los parámetros especificados.

```
bool  BuyStop(
   double                volume,                    // volumen de la orden
   double                price,                     // precio de la orden
   const string          symbol=NULL,               // símbolo
   double                sl=0.0,                    // precio stop loss
   double                tp=0.0,                    // precio take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // vida de la orden
   datetime              expiration=0,              // hora de expiración de la orc
   const string          comment=""                 // comentario
   )
```

Parámetros

volume

[in] Volumen de la orden.

price

[in] Precio de la orden.

symbol=NULL

[in] Símbolo de la orden. Si el símbolo no se especifica, se utiliza el símbolo actual.

sl=0.0

[in] Precio Stop Loss.

tp=0.0

[in] Precio Take Profit.

type_time=ORDER_TIME_GTC

[in] Vida de la orden (valor de la enumeración ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Hora de expiración de la orden (solo se usa si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Comentario de la orden.

Valor devuelto

true - si las estructuras se pueden comprobar correctamente; en caso contrario - false.

Nota

Si se completa correctamente el método BuyStop(...), esto no implica siempre una ejecución exitosa de la operación de trading. Es necesario comprobar el resultado de la petición de trading (el código de retorno del servidor) con el valor de ResultRetcode(), y el valor devuelto por ResultOrder().

© 2000-2025, MetaQuotes Ltd.

SellLimit

Coloca la orden pendiente de tipo Sell Limit (vender en el precio, más alto que el precio actual del mercado) con los parámetros especificados.

```
bool  SellLimit(
   double                volume,                    // volumen de la orden
   double                price,                     // precio de la orden
   const string          symbol=NULL,               // símbolo
   double                sl=0.0,                    // precio stop loss
   double                tp=0.0,                    // precio take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // vida de la orden
   datetime              expiration=0,              // hora de expiración de la orc
   const string          comment=""                 // comentario
   )
```

Parámetros

volume

[in] Volumen de la orden.

price

[in] Precio de la orden.

symbol=NULL

[in] Símbolo de la orden. Si el símbolo no se especifica, se utiliza el símbolo actual.

sl=0.0

[in] Precio Stop Loss.

tp=0.0

[in] Precio Take Profit.

type_time=ORDER_TIME_GTC

[in] Vida de la orden (valor de la enumeración ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Hora de expiración de la orden (solo se usa si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Comentario de la orden.

Valor devuelto

true - si las estructuras se pueden comprobar correctamente; en caso contrario - false.

Nota

Si se completa correctamente el método SellLimit(...), esto no siempre implica una ejecución exitosa de la operación de trading. Es necesario comprobar el resultado de la petición de trading (el código de retorno del servidor) con el valor de ResultRetcode(), y el valor devuelto por ResultOrder().

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

SellStop

Coloca la orden pendiente de tipo Sell Stop (vender en el precio, más bajo que el precio actual del mercado) con los parámetros especificados.

```
bool  SellStop(
   double                volume,                    // volumen de la orden
   double                price,                     // precio de la orden
   const string          symbol=NULL,               // símbolo
   double                sl=0.0,                    // precio stop loss
   double                tp=0.0,                    // precio take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // vida de la orden
   datetime              expiration=0,              // hora de expiración de la orc
   const string          comment=""                 // comentario
   )
```

Parámetros

volume

[in] Volumen de la orden.

price

[in] Precio de la orden.

symbol=NULL

[in] Símbolo de la orden. Si el símbolo no se especifica, se utiliza el símbolo actual.

sl=0.0

[in] Precio Stop Loss.

tp=0.0

[in] Precio Take Profit.

type_time=ORDER_TIME_GTC

[in] Vida de la orden (valor de la enumeración ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Hora de expiración de la orden (solo se usa si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Comentario de la orden.

Valor devuelto

true - si las estructuras se pueden comprobar correctamente; en caso contrario - false.

Nota

Si se completa correctamente el método SellStop(...), esto no siempre implica una ejecución exitosa de la operación de trading. Es necesario comprobar el resultado de la petición de trading (el código de retorno del servidor) con el valor de ResultRetcode(), y el valor devuelto por ResultOrder().

© 2000-2025, MetaQuotes Ltd.

Request

Obtiene la copia de la estructura de la última petición.

```
void  Request(
   MqlTradeRequest&  request      // estructura objetivo
   ) const
```

Parámetros

request

[out] Referencia a la estructura de tipo MqlTradeRequest.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

RequestAction

Obtiene el tipo de operación de trading.

```
ENUM_TRADE_REQUEST_ACTIONS  RequestAction() const
```

Valor devuelto

Tipo de operación de trading, utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestActionDescription

Obtiene el tipo de operación de trading en formato string.

```
string  RequestActionDescription() const
```

Valor devuelto

Tipo de operación de trading (en formato string), utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestMagic

Obtiene el número mágico del Asesor Experto.

```
ulong  RequestMagic() const
```

Valor devuelto

Número mágico (ID) del Asesor Experto, utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestOrder

Obtiene el ticket de la orden, utilizado en la última petición.

```
ulong  RequestOrder() const
```

Valor devuelto

Ticket de la orden de la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestSymbol

Obtiene el nombre del símbolo utilizado en la última petición.

```
string  RequestSymbol() const
```

Valor devuelto

Nombre del símbolo utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestVolume

Obtiene el volumen de trading (en lotes), utilizado en la última petición.

```
double  RequestVolume() const
```

Valor devuelto

Volumen de trading (en lotes), utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestPrice

Obtiene el precio utilizado en la última petición.

```
double  RequestPrice() const
```

Valor devuelto

Precio de la orden, utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestStopLimit

Obtiene el precio de la orden pendiente de tipo Stop Limit, utilizado en la última petición.

```
double  RequestStoplimit() const
```

Valor devuelto

Precio de la orden pendiente de tipo Stop Limit, utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestSL

Obtiene el precio de Stop Loss de la orden, utilizado en la última petición.

```
double  RequestSL() const
```

Valor devuelto

Precio de Stop Loss, utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestTP

Obtiene el precio de Take Profit de la orden, utilizado en la última petición.

```
double  RequestTP() const
```

Valor devuelto

Precio de Take Profit, utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestDeviation

Obtiene la desviación del precio de la orden, utilizado en la última petición.

```
ulong  RequestDeviation() const
```

Valor devuelto

La desviación del precio de la orden, utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestType

Obtiene el tipo de orden, utilizada en la última petición.

```
ENUM_ORDER_TYPE  RequestType() const
```

Valor devuelto

Tipo de orden, utilizada en la última petición (valor de la enumeración ENUM_ORDER_TYPE).

© 2000-2025, MetaQuotes Ltd.

RequestTypeDescription

Obtiene el tipo de orden, en formato string, utilizada en la última petición.

```
string  RequestTypeDescription() const
```

Valor devuelto

Tipo de orden, en formato string, utilizada en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestTypeFilling

Obtiene el tipo de relleno de la orden, utilizada en la última petición.

```
ENUM_ORDER_TYPE_FILLING  RequestTypeFilling() const
```

Valor devuelto

Tipo de relleno de la orden (valor de ENUM_ORDER_TYPE_FILLING), utilizada en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestTypeFillingDescription

Obtiene el tipo de relleno de la orden (en formato cadena) utilizada en la última petición.

```
string  RequestTypeFillingDescription() const
```

Valor devuelto

Tipo de relleno (en formato cadena) de la orden, utilizada en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestTypeTime

Obtiene el período de validez de la orden, utilizada en la última petición.

```
ENUM_ORDER_TYPE_TIME  RequestTypeTime() const
```

Valor devuelto

Período de validez de la orden (valor de la enumeración ENUM_ORDER_TYPE_TIME), utilizada en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestTypeTimeDescription

Obtiene el período de validez de la orden (como string), utilizada en la última petición.

```
string  RequestTypeTimeDescription() const
```

Valor devuelto

Período de validez de la orden (como string), utilizada en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestExpiration

Obtiene la hora de expiración de la orden, utilizada en la última petición.

```
datetime  RequestExpiration() const
```

Valor devuelto

Hora de expiración de la orden, utilizada en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestComment

Obtiene el comentario de la orden, utilizado en la última petición.

```
string  RequestComment() const
```

Valor devuelto

Comentario de la orden, utilizado en la última petición.

© 2000-2025, MetaQuotes Ltd.

RequestPosition

Obtiene el ticket de la posición.

```
ulong  RequestPosition() const
```

Valor devuelto

Ticket de la posición que se ha usado en la última solicitud.

© 2000-2025, MetaQuotes Ltd.

RequestPositionBy

Obtiene el ticket de la posición opuesta.

```
ulong  RequestPositionBy() const
```

Valor devuelto

Ticket de la posición opuesta que se ha usado en la última solicitud.

© 2000-2025, MetaQuotes Ltd.

Result

Obtiene la copia de la estructura del resultado de la última petición.

```
void  Result(
   MqlTradeResult&  result      // referencia
   ) const
```

Parámetros

result

[out] Referencia a la estructura de tipo MqlTradeResult.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

ResultRetcode

Obtiene el código del resultado de la petición.

```
uint  ResultRetcode() const
```

Valor devuelto

El código del resultado de la petición.

© 2000-2025, MetaQuotes Ltd.

ResultRetcodeDescription

Obtiene el código del resultado de la petición en formato texto.

```
string  ResultRetcodeDescription() const
```

Valor devuelto

Código del resultado de la última petición en formato texto.

© 2000-2025, MetaQuotes Ltd.

ResultDeal

Obtiene el ticket de la transacción.

```
ulong  ResultDeal() const
```

Valor devuelto

Ticket de la transacción, si la transacción se ejecuta.

© 2000-2025, MetaQuotes Ltd.

ResultOrder

Obtiene el ticket de la orden.

```
ulong  ResultOrder() const
```

Valor devuelto

Ticket de la orden, si la orden se coloca.

© 2000-2025, MetaQuotes Ltd.

ResultVolume

Obtiene el volumen de la transacción o de la orden.

```
double  ResultVolume() const
```

Valor devuelto

Volumen de la transacción o de la orden.

© 2000-2025, MetaQuotes Ltd.

ResultPrice

Obtiene el precio, confirmado por el broker.

```
double  ResultPrice() const
```

Valor devuelto

Precio, confirmado por el broker.

© 2000-2025, MetaQuotes Ltd.

ResultBid

Obtiene el precio bid actual (la recotización).

```
double  ResultBid() const
```

Valor devuelto

Precio bid actual (la recotización).

© 2000-2025, MetaQuotes Ltd.

ResultAsk

Obtiene el precio ask actual (la recotización).

```
double  ResultAsk() const
```

Valor devuelto

Precio ask actual (la recotización).

© 2000-2025, MetaQuotes Ltd.

ResultComment

Obtiene el comentario del broker.

```
string  ResultComment() const
```

Valor devuelto

Comentario del broker.

© 2000-2025, MetaQuotes Ltd.

CheckResult

Obtiene la copia de la estructura de la última solicitud de comprobación del resultado.

```
void  CheckResult(
   MqlTradeCheckResult&  check_result      // referencia
   ) const
```

Parámetros

check_result

[out] Referencia a la estructura objetivo de tipo MqlTradeCheckResult.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

CheckResultRetcode

Obtiene el valor del campo retcode de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

```
uint  CheckResultRetcode() const
```

Valor devuelto

El valor del campo retcode (código de error) de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

© 2000-2025, MetaQuotes Ltd.

CheckResultRetcodeDescription

Obtiene la descripción string del campo retcode de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

```
string  ResultRetcodeDescription() const
```

Valor devuelto

La descripción string del campo retcode (código de error) de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

© 2000-2025, MetaQuotes Ltd.

CheckResultBalance

Obtiene el valor del campo balance de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

```
double  CheckResultBalance() const
```

Valor devuelto

El valor del campo balance (después de la ejecución de la operación) de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

© 2000-2025, MetaQuotes Ltd.

CheckResultEquity

Obtiene el valor del campo equity de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

```
double  CheckResultEquity() const
```

Valor devuelto

El valor del campo equity (después de la ejecución de la operación) de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

© 2000-2025, MetaQuotes Ltd.

CheckResultProfit

Obtiene el valor del campo profit de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

```
double  CheckResultProfit() const
```

Valor devuelto

El valor del campo profit (después de la ejecución de la operación) de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

© 2000-2025, MetaQuotes Ltd.

CheckResultMargin

Obtiene el valor del campo margin de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

```
double  CheckResultMargin() const
```

Valor devuelto

El valor del campo margin (requerido por la operación de trading) de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

© 2000-2025, MetaQuotes Ltd.

CheckResultMarginFree

Obtiene el valor del campo margin_free de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

```
double  CheckResultMarginFree() const
```

Valor devuelto

El valor del campo margin_free (después de la ejecución de la operación) de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

© 2000-2025, MetaQuotes Ltd.

CheckResultMarginLevel

Obtiene el valor del campo margin_level de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

```
double  CheckResultMarginLevel() const
```

Valor devuelto

El valor del campo margin_level (después de la ejecución de la operación) de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

© 2000-2025, MetaQuotes Ltd.

CheckResultComment

Obtiene el valor del campo field de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

```
string  CheckResultComment() const
```

Valor devuelto

El valor del campo comment (comentario del código de respuesta, descripción del error) de tipo MqlTradeCheckResult, cumplimentado mientras se comprueba que la petición es correcta.

© 2000-2025, MetaQuotes Ltd.

PrintRequest

Imprime los parámetros de la última posición.

```
void  PrintRequest() const
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

PrintResult

Imprime los resultados de la última petición.

```
void  PrintResult() const
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

FormatRequest

Prepara la cadena con formato con los parámetros de la última petición.

```
string  FormatRequest(
   string&                 str,          // string objetivo
   const MqlTradeRequest&  request       // petición
   ) const
```

Parámetros

str

[in] string objetivo, pasada por referencia.

request

[in] Estructura de tipo MqlTradeRequest con los parámetros de la última petición.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

FormatRequestResult

Prepara la cadena con formato con los resultados de ejecución de la última solicitud.

```
string  FormatRequestResult(
   string&                 str,          // string
   const MqlTradeRequest&  request,      // estructura de la solicitud
   const MqlTradeResult&   result        // estructura resultado
   ) const
```

Parámetros

str

[in] string objetivo, pasada por referencia.

request

[in] Estructura de tipo MqlTradeRequest con los parámetros de la última petición.

result

[in] Estructura de tipo MqlTradeResult con los resultados de la última petición.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

CTerminalInfo

La clase CTerminalInfo facilita el acceso a las propiedades del entorno del programa mql5.

Descripción

La clase CTerminalInfo proporciona acceso a las propiedades del entorno del programa mql5.

Declaración

```
class CTerminalInfo : public CObject
```

Título

```
#include <Trade\TerminalInfo.mqh>
```

Jerarquía de herencia

CObject

CTerminalInfo

Métodos de la clase por grupos

Métodos de acceso a las propiedades de tipo integer	
Build	Obtiene el número de compilación del terminal cliente
IsConnected	Obtiene información sobre la conexión al servidor de trading
IsDLLsAllowed	Obtiene información acerca del permiso para utilizar DLL
IsTradeAllowed	Obtiene información sobre los permisos de trading
IsEmailEnabled	Obtiene información de los permisos para enviar e-mails al servidor SMTP, e inicio de sesión, especificado en la configuración del terminal
IsFtpEnabled	Obtiene información sobre los permisos para enviar informes comerciales al servidor FTP, e inicio de sesión, especificado en la configuración del terminal
MaxBars	Obtiene información sobre el número máximo de barras del gráfico
CodePage	Obtiene información sobre la página de código del idioma del terminal cliente
CPUCores	Obtiene información sobre los núcleos de la CPU
MemoryPhysical	Obtiene información sobre la memoria física (en Mb)
MemoryTotal	Obtiene información sobre la memoria total disponible en el proceso del terminal/agente (en Mb)

© 2000-2025, MetaQuotes Ltd.

Métodos de acceso a las propiedades de tipo integer	
MemoryAvailable	Obtiene información sobre la memoria libre disponible en el proceso del terminal/agente (en Mb)
MemoryUsed	Obtiene información sobre la memoria utilizada por el proceso del terminal/agente (en Mb)
IsX64	Obtiene información sobre el tipo de terminal cliente (32/64 bit)
OpenCLSupport	Obtiene información sobre la versión de OpenCL soportada por la tarjeta de vídeo
DiskSpace	Obtiene información sobre el espacio libre en el disco (en Mb)
Métodos de acceso a las propiedades de tipo string	
Language	Obtiene el idioma del terminal cliente
Name	Obtiene el nombre del terminal cliente
Company	Obtiene la compañía del terminal cliente
Path	Obtiene la carpeta del terminal cliente
DataPath	Obtiene la carpeta de datos del terminal cliente
CommonDataPath	Obtiene la carpeta de datos común de todos los terminales cliente instalados en la computadora
Acceso a las funciones del API de MQL5	
InfoInteger	Obtiene el valor de la propiedad de tipo integer
InfoString	Obtiene el valor de la propiedad de tipo string

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Build

Obtiene el número de compilación del terminal cliente.

```
int  CBuild() const
```

Valor devuelto

Número de compilación del terminal cliente.

Nota

Para obtener el número de compilación utiliza la función TerminalInfoInteger() (propiedad TERMINAL_BUILD).

© 2000-2025, MetaQuotes Ltd.

IsConnected

Obtiene información sobre la conexión al servidor de trading.

```
bool  IsConnected() const
```

Valor devuelto

true, si el terminal está conectado al servidor de trading; en caso contrario false.

Nota

Para obtener el estado de la conexión se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_CONNECTED).

© 2000-2025, MetaQuotes Ltd.

IsDLLsAllowed

Obtiene información acerca del permiso para utilizar DLL.

```
bool  IsDLLsAllowed() const
```

Valor devuelto

true, si el uso de DLL está permitido; en caso contrario false.

Nota

Para obtener el permiso de uso de DLL se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_DLLS_ALLOWED).

© 2000-2025, MetaQuotes Ltd.

IsTradeAllowed

Obtiene información sobre los permisos de trading.

```
bool  IsTradeAllowed() const
```

Valor devuelto

true, si el trading está permitido; en caso contrario, false.

Nota

Para obtener los permisos de trading se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_TRADE_ALLOWED).

© 2000-2025, MetaQuotes Ltd.

IsEmailEnabled

Obtiene información de los permisos para enviar e-mails al servidor SMTP, e inicio de sesión, especificado en la configuración del terminal.

```
bool  IsEmailEnabled() const
```

Valor devuelto

true, si el envío de emails está permitido; en caso contrario, false.

Nota

Para obtener el permiso de envío de emails se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_EMAIL_ENABLED).

© 2000-2025, MetaQuotes Ltd.

IsFtpEnabled

Obtiene información sobre los permisos para enviar informes comerciales al servidor FTP, e inicio de sesión, especificado en la configuración del terminal.

```
bool  IsFtpEnabled() const
```

Valor devuelto

true, si el envío de informes comerciales al servidor FTP está permitido; en caso contrario, false.

Nota

Para obtener información sobre los permisos para enviar los informes se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_FTP_ENABLED).

© 2000-2025, MetaQuotes Ltd.

MaxBars

Obtiene información sobre el número máximo de barras del gráfico, especificado en la configuración del terminal cliente.

```
int  MaxBars() const
```

Valor devuelto

Número máximo de barras del gráfico.

Nota

Para obtener el número máximo de barras del gráfico se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_MAXBARS).

© 2000-2025, MetaQuotes Ltd.

CodePage

Obtiene información sobre la página de código del idioma del terminal cliente.

```
int  CodePage() const
```

Valor devuelto

Página de código del idioma del terminal cliente.

Nota

Para obtener la página de código utiliza la función TerminalInfoInteger() (propiedad TERMINAL_CODEPAGE).

© 2000-2025, MetaQuotes Ltd.

CPUCores

Obtiene información sobre el número de núcleos de la CPU.

```
int  CPUCores() const
```

Valor devuelto

Número de núcleos de la CPU.

Nota

Para obtener la cantidad de núcleos de la CPU utiliza la función TerminalInfoInteger() (propiedad TERMINAL_CPU_CORES).

© 2000-2025, MetaQuotes Ltd.

MemoryPhysical

Obtiene información sobre la memoria física (en Mb).

```
int  MemoryPhysical() const
```

Valor devuelto

Memoria física (en Mb).

Nota

Para obtener la memoria física se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_MEMORY_PHYSICAL).

© 2000-2025, MetaQuotes Ltd.

MemoryTotal

Obtiene información sobre la memoria total disponible en el terminal/agente (en Mb).

```
int  MemoryTotal() const
```

Valor devuelto

Memoria total disponible en el terminal/agente (en Mb).

Nota

Para obtener la memoria total se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_MEMORY_TOTAL).

© 2000-2025, MetaQuotes Ltd.

MemoryAvailable

Obtiene información sobre la memoria libre disponible en el proceso del terminal/agente (en Mb).

```
int  MemoryTotal() const
```

Valor devuelto

Memoria libre disponible en el proceso del terminal/agente.

Nota

Para obtener la memoria libre se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_MEMORY_TOTAL).

© 2000-2025, MetaQuotes Ltd.

MemoryUsed

Obtiene información sobre la memoria utilizada por el terminal cliente/agente (en Mb).

```
int  MemoryUsed() const
```

Valor devuelto

Memoria utilizada por el terminal cliente/agente (en Mb).

Nota

Para obtener la memoria se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_MEMORY_USED).

© 2000-2025, MetaQuotes Ltd.

IsX64

Obtiene información sobre el tipo de terminal cliente.

```
bool  IsX64() const
```

Valor devuelto

true, si se utiliza una versión 64-bit; en caso contrario, false.

Nota

Para obtener el tipo de terminal cliente se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_X64).

© 2000-2025, MetaQuotes Ltd.

OpenCLSupport

Obtiene información sobre la versión de OpenCL soportada por la tarjeta de vídeo.

```
int  OpenCLSupport() const
```

Valor devuelto

El valor devuelto tiene esta forma: 0x00010002 = "1.2". El 0 significa que OpenCL no está soportado.

Nota

Para obtener la versión de OpenCL se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_OPENCL_SUPPORT).

© 2000-2025, MetaQuotes Ltd.

DiskSpace

Obtiene información sobre el espacio libre en el disco disponible en el terminal cliente/agente (en Mb).

```
int  MDiskSpace() const
```

Valor devuelto

Espacio libre en el disco disponible en el terminal cliente/agente (archivos guardados en la carpeta MQL5\Files).

Nota

Para obtener el espacio libre en el disco se utiliza la función TerminalInfoInteger() (propiedad TERMINAL_DISK_SPACE).

© 2000-2025, MetaQuotes Ltd.

Language

Obtiene el idioma del terminal cliente.

```
string  Language() const
```

Valor devuelto

Idioma utilizado en el terminal cliente.

Nota

Para obtener el idioma se utiliza la función TerminalInfoString() (propiedad TERMINAL_LANGUAGE).

© 2000-2025, MetaQuotes Ltd.

Name

Obtiene el nombre del terminal cliente.

```
string  Name() const
```

Valor devuelto

Nombre del terminal cliente.

Nota

Para obtener el nombre del terminal cliente se utiliza la función TerminalInfoString() (propiedad TERMINAL_NAME).

© 2000-2025, MetaQuotes Ltd.

Company

Obtiene información sobre el nombre del broker.

```
string  Company() const
```

Valor devuelto

Nombre del broker.

Nota

Para obtener el nombre del broker utiliza la función TerminalInfoString() (propiedad TERMINAL_COMPANY).

© 2000-2025, MetaQuotes Ltd.

Path

Obtiene la carpeta del terminal cliente.

```
string  Path() const
```

Valor devuelto

La carpeta del terminal cliente.

Nota

Para obtener la carpeta del terminal cliente se utiliza la función TerminalInfoString() (propiedad TERMINAL_PATH).

© 2000-2025, MetaQuotes Ltd.

DataPath

Obtiene información sobre la carpeta de datos del terminal cliente.

```
string  DataPath() const
```

Valor devuelto

Carpeta de datos del terminal cliente.

Nota

Para obtener la carpeta de datos del terminal cliente utiliza la función TerminalInfoString() (propiedad TERMINAL_DATA_PATH).

© 2000-2025, MetaQuotes Ltd.

CommonDataPath

Obtiene la carpeta de datos común de todos los terminales cliente instalados en la computadora.

```
string  CommonDataPath() const
```

Valor devuelto

Carpeta de datos común.

Nota

Para obtener la carpeta de datos utiliza la función TerminalInfoString() (propiedad COMMON_DATA_PATH).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Devuelve el valor de la propiedad correspondiente del entorno del programa mql5.

```
int  TerminalInfoInteger(
   int  property_id      // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad. Puede ser cualquiera de los valores de la enumeración ENUM_TERMINAL_INFO_INTEGER.

Valor devuelto

Valor de tipo int.

Nota

Para obtener el valor de la propiedad se utiliza la función TerminalInfoInteger().

© 2000-2025, MetaQuotes Ltd.

InfoString

La función devuelve el valor de la propiedad correspondiente del entorno del programa mql5. La propiedad tiene que ser de tipo string.

```
string  TerminalInfoString(
   int  property_id      // identificador de la propiedad
   );
```

Parámetros

property_id

[in] Identificador de la propiedad. Uno de los valores de la enumeración ENUM_TERMINAL_INFO_STRING.

Valor devuelto

Valor de tipo string.

Nota

Para obtener el valor de la propiedad se utiliza la función TerminalInfoString().

© 2000-2025, MetaQuotes Ltd.

Clases de trading strategy

Esta sección contiene detalles técnicos que explican cómo trabajar con clases para crear y probar estrategias de trading, así como una descripción relevante de los componentes de la librería estándar de MQL5.

Utilizar estas clases le ahorrará tiempo al crear sus estrategias de trading.

La librería estándar MQL5 (en lo que a estrategias de trading se refiere) se encuentra en la carpeta Include\Expert del terminal.

Base classes	Descripción
CExpertBase	Clase base de todas las estrategias de trading
CExpert	Clase base de los asesores expertos
CExpertSignal	Clase base de las señales de trading
CExpertTrailing	Clase base de los Trailing Stop
CExpertMoney	Clase base de las clases de gestión del dinero

Trading signal classes	Descripción
CSignalAC	El módulo de las señales basadas en los modelos de mercado del indicador Accelerator Oscillator.
CSignalAMA	El módulo de las señales basadas en los modelos de mercado del indicador Media Móvil Adaptativa.
CSignalAO	El módulo de las señales basadas en los modelos de mercado del indicador Awesome Oscillator.
CSignalBearsPower	El módulo de las señales basadas en los modelos de mercado del oscilador Bears Power.
CSignalBullsPower	El módulo de las señales basadas en los modelos de mercado del oscilador Bulls Power.
CSignalCCI	El módulo de las señales basadas en los modelos de mercado del oscilador Commodity Channel Index.
CSignalDeM	El módulo de las señales basadas en los modelos de mercado del oscilador DeMarker.
CSignalDEMA	El módulo de las señales basadas en los modelos de mercado del indicador Media Móvil Exponencial Doble.
CSignalEnvelopes	El módulo de las señales basadas en los modelos de mercado del indicador Envelopes.
CSignalFrAMA	El módulo de las señales basadas en los modelos de mercado del indicador Media Móvil Adaptativa Fractal.
CSignalITF	El módulo de filtrado de señales por tiempo.

© 2000-2025, MetaQuotes Ltd.

Trading signal classes	Descripción
CSignalMACD	El módulo de las señales basadas en los modelos de mercado del oscilador MACD.
CSignalMA	El módulo de las señales basadas en los modelos de mercado del indicador Media Móvil.
CSignalSAR	El módulo de las señales basadas en los modelos de mercado del indicador SAR Parabólico.
CSignalRSI	El módulo de las señales basadas en los modelos de mercado del oscilador Índice de Fuerza Relativa.
CSignalRVI	El módulo de las señales basadas en los modelos de mercado del oscilador Índice de Vigor Relativo.
CSignalStoch	El módulo de las señales basadas en los modelos de mercado del Oscilador Estocástico.
CSignalTRIX	El módulo de las señales basadas en los modelos de mercado del oscilador Media Exponencial Triple.
CSignalTEMA	El módulo de las señales basadas en los modelos de mercado del indicador Media Móvil Exponencial Triple.
CSignalWPR	El módulo de las señales basadas en los modelos de mercado del oscilador Rango Porcentual de Williams.

Trailing Stop classes	Descripción
CTrailingFixedPips	Esta clase implementa un algoritmo de Trailing Stop basado en puntos fijos
CTrailingMA	Esta clase implementa un algoritmo de Trailing Stop basado en los valores del indicador Media Móvil
CTrailingNone	Una clase stub que no utiliza ningún algoritmo de Trailing Stop
CTrailingPSAR	Esta clase implementa un algoritmo de Trailing Stop basado en los valores del indicador SAR Parabólico

Clases de gestión del dinero	Descripción
CMoneyFixedLot	Clase con un algoritmo de trading basado en el tamaño predefinido de lote fijo.
CMoneyFixedMargin	Clase con un algoritmo de trading basado en el margen fijo predefinido.
CMoneyFixedRisk	Clase con un algoritmo de trading basado en el riesgo predefinido.
CMoneyNone	Clase con un algoritmo de trading basado en el tamaño de lote mínimo permitido.

© 2000-2025, MetaQuotes Ltd.

Clases de gestión del dinero	Descripción
CMoneySizeOptimized	Clase con un algoritmo de trading basado en el el tamaño de lote variable que depende de los resultados de las operaciones anteriores.

© 2000-2025, MetaQuotes Ltd.

Clases base de los Asesores Expertos

Esta sección contiene detalles técnicos que explican cómo trabajar con clases para crear y probar estrategias de trading, así como una descripción relevante de los componentes de la librería estándar de MQL5.

Utilizar estas clases le ahorrará tiempo al crear sus estrategias de trading.

La librería estándar MQL5 (en lo que a estrategias de trading se refiere) se encuentra en la carpeta Include\Expert del terminal.

Clase	Descripción
CExpertBase	Clase base de todas las estrategias de trading
CExpert	Clase base de los asesores expertos
CExpertSignal	Clase base de las señales de trading
CExpertTrailing	Clase base de los Trailing Stop
CExpertMoney	Clase base de las clases de gestión del dinero

© 2000-2025, MetaQuotes Ltd.

CExpertBase

CExpertBase es la clase base de la clase CExpert y de todas las clases de estrategias de trading.

Descripción

CExpertBase proporciona los datos y los métodos comunes a todos los objetos del Asesor Experto.

Declaración

```
class CExpertBase : public CObject
```

Título

```
#include <Expert\ExpertBase.mqh>
```

Jerarquía de herencia

CObject

CExpertBase

Descendientes directos

CExpert, CExpertMoney, CExpertSignal, CExpertTrailing

Métodos de la clase

Métodos públicos:

Inicialización	
virtual Init	Método de inicialización de la instancia de la clase
virtual ValidationSettings	Comprueba las configuraciones
Parámetros	
Symbol	Establece el símbolo
Period	Establece el periodo de tiempo
Magic	Establece el ID del Asesor Experto
Indicadores y series temporales	
virtual SetPriceSeries	Establece los punteros a las series temporales externas (series de precios)
virtual SetOtherSeries	Establece los punteros a las series temporales externas (series)
virtual InitIndicators	Inicializa los indicadores y las series temporales
Acceso a datos protegidos	
InitPhase	Obtiene la fase actual de inicialización del objeto

© 2000-2025, MetaQuotes Ltd.

Inicialización	
TrendType	Establece el tipo de tendencia
UsedSeries	Obtiene la máscara de bits de la serie temporal utilizada
EveryTick	Establece la bandera "Cada tick"
Acceso a las series temporales	
Open	Obtiene el elemento de la serie temporal Open por índice.
High	Obtiene el elemento de la serie temporal High por índice
Low	Obtiene el elemento de la serie temporal Low por índice
Close	Obtiene el elemento de la serie temporal Close por índice
Spread	Obtiene el elemento de la serie temporal Spread por índice
Time	Obtiene el elemento de la serie temporal Time por índice
TickVolume	Obtiene el elemento de la serie temporal TickVolume por índice
RealVolume	Obtiene el elemento de la serie temporal RealVolume por índice

Protected Methods:

Inicialización de series temporales	
InitOpen	Método de inicialización de la serie temporal Open
InitHigh	Método de inicialización de la serie temporal High
InitLow	Método de inicialización de la serie temporal Low
InitClose	Método de inicialización de la serie temporal Close
InitSpread	Método de inicialización de la serie temporal Spread
InitTime	Método de inicialización de la serie temporal Time
InitTickVolume	Método de inicialización de la serie temporal TickVolume
InitRealVolume	Método de inicialización de la serie temporal RealVolume
Métodos de servicio	
virtual PriceLevelUnit	Obtiene la unidad del nivel del precio
virtual StartIndex	Obtiene el índice de la barra inicial a analizar
virtual CompareMagic	Compara el ID del Asesor Experto con el valor especificado

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

InitPhase

Obtiene la fase actual de la inicialización del objeto.

```
ENUM_INIT_PHASE  InitPhase()
```

Valor devuelto

Fase actual de la inicialización del objeto.

Nota

El objeto se inicializa en varias fases:

1. Comienza la inicialización.

- inicio - después de finalizar el constructor
- fin - después de que el método Init(...) termine de ejecutarse correctamente.
- permitido - llamada al método Init(...)
- no permitido - llamada al método ValidationSettings() y otros métodos de inicialización

2. Fase de establecimiento de parámetros. En esta fase es necesario establecer los parámetros del objeto utilizados en la creación de indicadores.

- inicio - después de que el método Init(...) termine de ejecutarse correctamente
- fin - después de que el método ValidationSettings() termine de ejecutarse correctamente
- permitido - llamada a los métodos Symbol(...) y Period(...)
- no permitido - llamada a los métodos Init(...), SetPriceSeries(...), SetOtherSeries(...) y InitIndicators(...)

3. Comprobación de los parámetros.

- inicio - después de que el método ValidationSettings() termine de ejecutarse correctamente
- fin - después de que el método InitIndicators(...) termine de ejecutarse correctamente
- permitido - llamada a los métodos Symbol(...), Period(...) y InitIndicators(...)
- no permitido - llamada a cualquier otro método de inicialización

4. Fin de la inicialización.

- inicio - después de que el método InitIndicators(...) termine de ejecutarse correctamente
- no permitido - llamada de métodos de inicialización

© 2000-2025, MetaQuotes Ltd.

TrendType

Establece el tipo de tendencia.

```
void  TrendType(
   M_TYPE_TREND    value          // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo del tipo de tendencia.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

UsedSeries

Obtiene la máscara de bits de la serie temporal utilizada.

```
int  UsedSeries()
```

Valor devuelto

La lista de la serie temporal utilizada como máscara de bits.

Nota

Si el bit se establece, entonces se utiliza la serie temporal correspondiente; en caso contrario si no se establece, la serie temporal no se utiliza.

La correspondencia bit-serie temporal:

bit 0 - serie temporal Open,
bit 1 - serie temporal High,
bit 2 - serie temporal Low,
bit 3 - serie temporal Close,
bit 4 - serie temporal Spread,
bit 5 - serie temporal Time,
bit 6 - serie temporal TickVolume,
bit 7 - serie temporal RealVolume.

© 2000-2025, MetaQuotes Ltd.

EveryTick

Establece la bandera "Cada tick".

```
void  EveryTick(
   bool     value          // bandera
   )
```

Parámetros

value

[in] Valor nuevo de la bandera.

Valor devuelto

Ninguno.

Nota

Si la bandera no se establece, el método de procesamiento solo se llama en la barra nueva del periodo de tiempo y símbolo actuales.

© 2000-2025, MetaQuotes Ltd.

Open

Obtiene el elemento de la serie temporal Open por índice.

```
double  Open(
   int     ind          // índice
   )
```

Parámetros

ind

[in] Índice del elemento.

Valor devuelto

Si se ejecuta correctamente, devuelve el valor numérico del elemento de la serie temporal Open con el índice especificado, en caso contrario devuelve EMPTY_VALUE.

Nota

El valor EMPTY_VALUE se devuelve en dos casos:

1. La serie temporal no se utiliza (no se establece el bit correspondiente).
2. El índice del elemento está fuera del rango.

© 2000-2025, MetaQuotes Ltd.

High

Obtiene el elemento de la serie temporal High por índice.

```
double  High(
   int     ind          // índice
   )
```

Parámetros

ind

[in] Índice del elemento.

Valor devuelto

Si se ejecuta correctamente, devuelve el valor numérico del elemento de la serie temporal High con el índice especificado, en caso contrario devuelve EMPTY_VALUE.

Nota

El valor EMPTY_VALUE se devuelve en dos casos:

1. La serie temporal no se utiliza (no se establece el bit correspondiente).
2. El índice del elemento está fuera del rango.

© 2000-2025, MetaQuotes Ltd.

Low

Obtiene el elemento de la serie temporal Low por índice.

```
double  Low(
   int     ind          // índice
   )
```

Parámetros

ind

[in] Índice del elemento.

Valor devuelto

Si se ejecuta correctamente, devuelve el valor numérico del elemento de la serie temporal Low con el índice especificado, en caso contrario devuelve EMPTY_VALUE.

Nota

El valor EMPTY_VALUE se devuelve en dos casos:

1. La serie temporal no se utiliza (no se establece el bit correspondiente).
2. El índice del elemento está fuera del rango.

© 2000-2025, MetaQuotes Ltd.

Close

Obtiene el elemento de la serie temporal Close por índice.

```
double  Close(
   int    ind         // índice
   )
```

Parámetros

ind

[in] Índice del elemento.

Valor devuelto

Si se ejecuta correctamente, devuelve el valor numérico del elemento de la serie temporal Close con el índice especificado, en caso contrario devuelve EMPTY_VALUE.

Nota

El valor EMPTY_VALUE se devuelve en dos casos:

1. La serie temporal no se utiliza (no se establece el bit correspondiente).
2. El índice del elemento está fuera del rango.

© 2000-2025, MetaQuotes Ltd.

Spread

Obtiene el elemento de la serie temporal Spread por índice.

```
double  Spread(
   int     ind          // índice
   )
```

Parámetros

ind

[in] Índice del elemento.

Valor devuelto

Si se ejecuta correctamente, devuelve el valor numérico del elemento de la serie temporal Spread con el índice especificado, en caso contrario devuelve EMPTY_VALUE.

Nota

El valor EMPTY_VALUE se devuelve en dos casos:

1. La serie temporal no se utiliza (no se establece el bit correspondiente).
2. El índice del elemento está fuera del rango.

© 2000-2025, MetaQuotes Ltd.

Time

Obtiene el elemento de la serie temporal Time por índice.

```
datetime  Time(
   int    ind          // índice
   )
```

Parámetros

ind

[in] Índice del elemento.

Valor devuelto

Si se ejecuta correctamente, devuelve el valor numérico del elemento de la serie temporal Time con el índice especificado, en caso contrario devuelve EMPTY_VALUE.

Nota

El valor EMPTY_VALUE se devuelve en dos casos:

1. La serie temporal no se utiliza (no se establece el bit correspondiente).
2. El índice del elemento está fuera del rango.

© 2000-2025, MetaQuotes Ltd.

TickVolume

Obtiene el elemento de la serie temporal TickVolume por índice.

```
long  TickVolume(
   int    ind         // Índice
   )
```

Parámetros

ind

[in] Índice del elemento.

Valor devuelto

Si se ejecuta correctamente, devuelve el valor numérico del elemento de la serie temporal TickVolume con el índice especificado, en caso contrario devuelve EMPTY_VALUE.

Nota

El valor EMPTY_VALUE se devuelve en dos casos:

1. La serie temporal no se utiliza (no se establece el bit correspondiente).
2. El índice del elemento está fuera del rango.

© 2000-2025, MetaQuotes Ltd.

RealVolume

Obtiene el elemento de la serie temporal RealVolume por índice.

```
long  RealVolume(
   int    ind         // índice
   )
```

Parámetros

ind

[in] Índice del elemento.

Valor devuelto

Si se ejecuta correctamente, devuelve el valor numérico del elemento de la serie temporal RealVolume con el índice especificado, en caso contrario devuelve EMPTY_VALUE.

Nota

El valor EMPTY_VALUE se devuelve en dos casos:

1. La serie temporal no se utiliza (no se establece el bit correspondiente).
2. El índice del elemento está fuera del rango.

© 2000-2025, MetaQuotes Ltd.

Init

Inicializa el objeto.

```
bool  Init(
   CSymbolInfo      symbol,    // símbolo
   ENUM_TIMEFRAMES  period,    // periodo de tiempo
   double           point      // punto
   )
```

Parámetros

symbol

[in] Puntero al objeto de tipo CSymbolInfo para acceder a la información del símbolo.

period

[in] Periodo de tiempo (enumeración ENUM_TIMEFRAMES).

point

[in] El "peso" del punto de 2/4-dígitos.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Symbol

Establece el símbolo.

```
bool  Symbol(
   string    name     // símbolo
   )
```

Parámetros

name

[in] Símbolo.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La configuración del símbolo actual hace falta si el objeto utiliza otro símbolo diferente al definido en la inicialización.

© 2000-2025, MetaQuotes Ltd.

Period

Establece el periodo temporal.

```
bool  Period(
   ENUM_TIMEFRAMES  value     // periodo temporal
   )
```

Parámetros

value

[in] Periodo temporal.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

El ajuste del periodo de tiempo es necesario si el objeto utiliza uno diferente al definido en la inicialización.

© 2000-2025, MetaQuotes Ltd.

Magic

Establece el ID del Asesor Experto.

```
void  Magic(
   ulong  value     // magia
   )
```

Parámetros

value

[in] ID del Asesor Experto.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Comprueba las configuraciones.

```
virtual bool  ValidationSettings()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

SetPriceSeries

Establece los punteros a series externas de precios.

```
virtual bool  SetPriceSeries(
   CiOpen*    open,         // puntero
   CiHigh*    high,         // puntero
   CiLow*     low,          // puntero
   CiClose*   close         // puntero
   )
```

Parámetros

open

[in] Puntero a la serie temporal Open.

high

[in] Puntero a la serie temporal High.

low

[in] Puntero a la serie temporal Low.

close

[in] Puntero a la serie temporal Close.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La configuración de punteros a series temporales externas (series de precios) hace falta si el objeto utiliza series temporales del símbolo y del marco temporal diferentes a los definidos en la inicialización.

© 2000-2025, MetaQuotes Ltd.

SetOtherSeries

Establece los punteros a series externas (no precio).

```
virtual bool  SetOtherSeries(
   CiSpread*       spread,        // puntero
   CiTime*         time,          // puntero
   CiTickVolume*   tick_volume,   // puntero
   CiRealVolume*   real_volume    // puntero
   )
```

Parámetros

spread

[in] Puntero a la serie temporal Spread.

time

[in] Puntero a la serie temporal Time.

tick_volume

[in] Puntero a la serie temporal TickVolume.

real_volume

[in] Puntero a la serie temporal RealVolume.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La configuración de punteros a series temporales externas (no precio) hace falta si el objeto utiliza series temporales del símbolo y del marco temporal diferentes a los definidos en la inicialización.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

Inicializa todos los indicadores y series temporales.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La serie temporal solo se inicializa si el objeto utiliza un símbolo o periodo temporal diferentes a los definidos en la inicialización.

© 2000-2025, MetaQuotes Ltd.

InitOpen

Inicializa la serie temporal Open.

```
bool  InitOpen(
   CIndicators*  indicators    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La serie temporal Open solo se inicializa si el Asesor Experto utiliza un símbolo/periodo temporal diferentes a los definidos en la inicialización (y se utiliza más serie temporal).

© 2000-2025, MetaQuotes Ltd.

InitHigh

Inicializa la serie temporal High.

```
bool  InitHigh(
   CIndicators*  indicators    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La serie temporal High solo se inicializa si el Asesor Experto utiliza un símbolo/periodo temporal diferentes a los definidos en la inicialización (y se utiliza más serie temporal).

© 2000-2025, MetaQuotes Ltd.

InitLow

Inicializa la serie temporal Low.

```
bool  InitLow(
   CIndicators*  indicators    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La serie temporal Low solo se inicializa si el Asesor Experto utiliza un símbolo/periodo temporal diferentes a los definidos en la inicialización (y se utiliza más serie temporal).

© 2000-2025, MetaQuotes Ltd.

InitClose

Inicializa la serie temporal Close.

```
bool  InitClose(
   CIndicators*  indicators    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La serie temporal Close solo se inicializa si el Asesor Experto utiliza un símbolo/periodo temporal diferentes a los definidos en la inicialización (y se utiliza más serie temporal).

© 2000-2025, MetaQuotes Ltd.

InitSpread

Inicializa la serie temporal Spread.

```
bool  InitSpread(
   CIndicators*  indicators    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La serie temporal Spread solo se inicializa si el Asesor Experto utiliza un símbolo/periodo temporal diferentes a los definidos en la inicialización (y se utiliza más serie temporal).

© 2000-2025, MetaQuotes Ltd.

InitTime

Inicializa la serie temporal Time.

```
bool  InitTime(
   CIndicators*  indicators    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La serie temporal Time solo se inicializa si el Asesor Experto utiliza un símbolo/periodo temporal diferentes a los definidos en la inicialización (y se utiliza más serie temporal).

© 2000-2025, MetaQuotes Ltd.

InitTickVolume

Inicializa la serie temporal TickVolume.

```
bool  InitTickVolume(
   CIndicators*  indicators    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La serie temporal TickVolume solo se inicializa si el Asesor Experto utiliza un símbolo/periodo temporal diferentes a los definidos en la inicialización (y se utiliza más serie temporal).

© 2000-2025, MetaQuotes Ltd.

InitRealVolume

Inicializa la serie temporal RealVolume.

```
bool  InitRealVolume(
   CIndicators*  indicators    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La serie temporal RealVolume solo se inicializa si el Asesor Experto utiliza un símbolo/periodo temporal diferentes a los definidos en la inicialización (y se utiliza más serie temporal).

© 2000-2025, MetaQuotes Ltd.

PriceLevelUnit

Obtiene la unidad del nivel del precio.

```
virtual double  PriceLevelUnit()
```

Valor devuelto

El valor de la unidad del nivel del precio.

Nota

El método de la clase base devuelve el "peso" del punto de 2/4 dígitos.

© 2000-2025, MetaQuotes Ltd.

StartIndex

Obtiene el índice de la barra de inicio a analizar.

```
virtual int  StartIndex()
```

Valor devuelto

El índice de la barra de inicio a analizar.

Nota

El método devuelve 0 si la bandera para analizar la barra actual se establece a true (análisis de la barra actual). Si la bandera no se establece, devuelve 1 (análisis de la última barra completada).

© 2000-2025, MetaQuotes Ltd.

CompareMagic

Compara el ID del Asesor Experto (magia) con el valor especificado.

```
virtual bool  CompareMagic(
   ulong  magic    // valor a comparar
   )
```

Parámetros

magic

[in] Valor a comparar.

Valor devuelto

true si son iguales, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CExpert

CExpert es la clase base de las estrategias de trading. Incorpora algoritmos para trabajar con series temporales, indicadores, así como un conjunto de métodos virtuales de estrategias de trading.

Cómo utilizarla:

1. Preparar un algoritmo para la estrategia;
2. Cree su propia clase, haciéndola heredar de CExpert;
3. Sobreescriba los métodos virtuales de su clase, escribiendo sus propios algoritmos.

Descripción

La clase CExpert ofrece un conjunto de métodos virtuales para implementar estrategias de trading.

Observación

La posición se determina mediante la pareja de propiedades m_symbol y m_magic, que pertenecen al experto y lo controlan. En el modo "cobertura", en la cuenta pueden existir de forma simultánea varias posiciones del mismo símbolo, por eso hay que prestar especial atención al valor m_magic.

Declaración

```
class CExpert : public CExpertBase
```

Título

```
#include <Expert\Expert.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpert

Métodos de la clase

Inicialización	
Init	Método de inicialización de la instancia de la clase
virtual InitSignal	Inicializa el objeto de la señal de trading
virtual InitTrailing	Inicializa el objeto de Trailing Stop
virtual InitMoney	Inicializa el objeto de la gestión del dinero
virtual InitTrade	Inicializa el objeto Trade
virtual ValidationSettings	Comprueba las configuraciones
virtual InitIndicators	Inicializa los indicadores y las series temporales
virtual InitParameters	Método de inicialización de los parámetros

© 2000-2025, MetaQuotes Ltd.

Inicialización	
virtual Deinit	Método de desinicialización de la instancia de la clase
virtual DeinitSignal	Desinicializa el objeto de la señal de trading
virtual DeinitTrailing	Desinicializa el objeto de Trailing Stop
virtual DeinitMoney	Desinicializa el objeto de la gestión del dinero
virtual DeinitTrade	Desinicializa el objeto Trade
virtual DeinitIndicators	Desinicializa los indicadores y las series temporales
Parámetros	
Magic	Establece el ID del Asesor Experto
MaxOrders	Obtiene/Establece la cantidad máxima de órdenes permitidas
OnTickProcess	Establece la bandera del evento "OnTick"
OnTradeProcess	Establece la bandera del evento "OnTrade"
OnTimerProcess	Establece la bandera del evento "OnTimer"
OnChartEventProcess	Establece la bandera del evento "OnChartEvent"
OnBookEventProcess	Establece la bandera del evento "OnBookEvent"
Métodos de procesamiento de eventos	
OnTick	Manejador del evento OnTick
OnTrade	Manejador del evento OnTrade
OnTimer	Manejador del evento OnTimer
OnChartEvent	Manejador del evento OnChartEvent
OnBookEvent	Manejador del evento OnBookEvent
Métodos de actualización	
Refresh	Actualiza todos los datos
Processing	
Procesamiento	Algoritmo de procesamiento principal
Métodos de entrada al mercado	
CheckOpen	Comprueba las condiciones de apertura de la posición
CheckOpenLong	Comprueba las condiciones de apertura de la posición larga
CheckOpenShort	Comprueba las condiciones de apertura de la posición corta
OpenLong	Abre una posición larga
OpenShort	Abre una posición corta

© 2000-2025, MetaQuotes Ltd.

Inicialización	
Métodos de salida del mercado	
CheckClose	Comprueba las condiciones de cierre de la posición actual
CheckCloseLong	Comprueba las condiciones de cierre de la posición larga
CheckCloseShort	Comprueba las condiciones de cierre de la posición corta
CloseAll	Cierra la posición abierta y borra todas las órdenes
Close	Cierra la posición abierta
CloseLong	Cierra la posición larga
CloseShort	Cierra la posición corta
Métodos para revertir la posición	
CheckReverse	Comprueba las condiciones para revertir la posición abierta
CheckReverseLong	Comprueba las condiciones para revertir la posición larga
CheckReverseShort	Comprueba las condiciones para revertir la posición corta
ReverseLong	Realiza la operación inversa de la posición larga
ReverseShort	Realiza la operación inversa de la posición corta
Métodos de trailing de la posición/orden	
CheckTrailingStop	Comprueba las condiciones para modificar los parámetros de la posición
CheckTrailingStopLong	Comprueba las condiciones de Trailing Stop de la posición larga
CheckTrailingStopShort	Comprueba las condiciones de Trailing Stop de la posición corta
TrailingStopLong	Realiza el Trailing Stop de la posición larga
TrailingStopShort	Realiza el Trailing Stop de la posición corta
CheckTrailingOrderLong	Comprueba las condiciones de Trailing Stop de la orden pendiente Buy Limit/Stop
CheckTrailingOrderShort	Comprueba las condiciones de Trailing Stop de la orden pendiente Sell Limit/Stop
TrailingOrderLong	Realiza el Trailing Stop de la orden pendiente Buy Limit/Stop
TrailingOrderShort	Realiza el Trailing Stop de la orden pendiente Sell Limit/Stop
Métodos para borrar la orden	

© 2000-2025, MetaQuotes Ltd.

Inicialización	
CheckDeleteOrderLong	Comprueba las condiciones para borrar la orden pendiente de compra
CheckDeleteOrderShort	Comprueba las condiciones para borrar la orden pendiente de venta
DeleteOrders	Borra todas las órdenes
DeleteOrder	Borra la orden pendiente Stop/Limit
DeleteOrderLong	Borra la orden pendiente Buy Limit/Stop
DeleteOrderShort	Borra la orden pendiente Sell Limit/Stop
Métodos de volumen de la operación	
LotOpenLong	Obtiene el volumen de la operación de compra
LotOpenShort	Obtiene el volumen de la operación de venta
LotReverse	Obtiene el volumen de la operación inversa de la posición
Métodos del historial de operaciones	
PrepareHistoryDate	Establece la fecha de inicio del seguimiento del historial de trading
HistoryPoint	Crea un punto de control del historial de trading (guarda el número de posiciones, las órdenes y las transacciones)
CheckTradeState	Compara el estado actual con el guardado y llama al manejador de evento correspondiente
Banderas de eventos	
WaitEvent	Establece la bandera de espera del evento
NoWaitEvent	Reinicia la bandera de espera del evento
Métodos de procesamiento de eventos	
TradeEventPositionStopTake	Manejador del evento "Disparar el Stop Loss/Take Profit de la posición"
TradeEventOrderTriggered	Manejador del evento "Disparar orden pendiente"
TradeEventPositionOpened	Manejador del evento "Posición abierta"
TradeEventPositionVolumeChanged	Manejador del evento "Volumen de la posición cambiado"
TradeEventPositionModified	Manejador del evento "Posición modificada"
TradeEventPositionClosed	Manejador del evento "Posición cerrada"

© 2000-2025, MetaQuotes Ltd.

Inicialización	
TradeEventOrderPlaced	Manejador del evento "Orden pendiente colocada"
TradeEventOrderModified	Manejador del evento "Orden pendiente modificada"
TradeEventOrderDeleted	Manejador del evento "Orden pendiente borrada"
TradeEventNotIdentified	Manejador del evento no identificado
Métodos de servicio	
TimeframeAdd	Agrega un periodo de tiempo a seguir
TimeframesFlags	Obtiene la bandera que indica los periodos de tiempo con una barra nueva
SelectPosition	Elegir una posición para trabajar posteriormente con ella.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

Init

Método de inicialización de la instancia de la clase.

```
bool  Init(
   string              symbol,         // símbolo
   ENUM_TIMEFRAMES     period,         // marco temporal
   bool                every_tick,     // bandera
   ulong               magic           // magia
   )
```

Parámetros

symbol

[in] Símbolo.

period

[in] Marco temporal (enumeración ENUM_TIMEFRAMES).

every_tick

[in] Bandera

magic

[in] ID del Asesor Experto (Número mágico).

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

Si every_tick se establece a true, entonces el método Processing() se llama en cada tick del símbolo actual. De otro modo, el método Processing() solamente se llama en la barra nueva del símbolo actual.

© 2000-2025, MetaQuotes Ltd.

Magic

Establece el ID del Asesor Experto (magia).

```
void  Magic(
   ulong  value     // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo del ID del Asesor Experto.

Valor devuelto

Ninguno.

Nota

Establece el valor del ID del Asesor Experto (magia) en las siguientes clases: Trade, Signal, Money, Trailing.

Implementación

```
//+------------------------------------------------------------------+
//| Establece el número mágico del objeto y de sus objetos dependientes           |
//| ENTRADA:  value - nuevo valor del número mágico.                       |
//| SALIDA: no.                                                      |
//| OBSERVACIONES: no.                                                        |
//+------------------------------------------------------------------+
void CExpert::Magic(ulong value)
  {
   if(m_trade!=NULL)     m_trade.SetExpertMagicNumber(value);
   if(m_signal!=NULL)    m_signal.Magic(value);
   if(m_money!=NULL)     m_money.Magic(value);
   if(m_trailing!=NULL) m_trailing.Magic(value);
//---
   CExpertBase::Magic(value);
  }
```

© 2000-2025, MetaQuotes Ltd.

InitSignal

Inicializa el objeto Trading Signal.

```
virtual bool  InitSignal(
   CExpertSignal*     signal=NULL,        // puntero
   )
```

Parámetros

signal

[in] Puntero al objeto de la clase CExpertSignal (o clases derivadas).

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

Si signal es NULL, se utilizará la clase CExpertSignal, que no hace nada.

© 2000-2025, MetaQuotes Ltd.

InitTrailing

Inicializa el objeto Trailing Stop.

```
virtual bool  InitTrailing(
   CExpertTrailing*     trailing=NULL,        // puntero
   )
```

Parámetros

trailing

[in] Puntero al objeto de la clase CExpertTrailing (o clases derivadas).

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

Si trailing es NULL, se utilizará la clase ExpertTrailing, que no hace nada.

© 2000-2025, MetaQuotes Ltd.

InitMoney

Inicializa el objeto de gestión del dinero.

```
virtual bool  InitMoney(
   CExpertMoney*     money=NULL,         // puntero
   )
```

Parámetros

money

[in] Puntero al objeto de la clase CExpertMoney (o clases derivadas).

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

Si money es NULL, se utilizará la clase CExpertMoney , con el mínimo lote.

© 2000-2025, MetaQuotes Ltd.

InitTrade

Inicializa el objeto Trade.

```
virtual bool  InitTrade(
   ulong             magic,        // magia
   CExpertTrade*     trade=NULL    // puntero
   )
```

Parámetros

magic

[in] ID del Asesor Experto (utilizado en las solicitudes de trading).

trade

[in] Puntero al objeto CExpertTrade.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Deinit

Método de desinicialización de la instancia de la clase.

```
virtual void Deinit()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

OnTickProcess

Establece la bandera para procesar el evento OnTick.

```
void  OnTickOProcess(
   bool     value        // bandera
   )
```

Parámetros

value

[in] Bandera para procesar el evento OnTick .

Valor devuelto

Ninguno.

Nota

Si la bandera vale true, se procesa el evento OnTick, de forma predeterminada, la bandera vale true.

© 2000-2025, MetaQuotes Ltd.

OnTradeProcess

Establece la bandera para procesar el evento OnTrade .

```
void  OnTradeProcess(
   bool     value        // bandera
   )
```

Parámetros

value

[in] Bandera para procesar el evento OnTrade.

Valor devuelto

Ninguno.

Nota

Si la bandera es true, se procesa el evento OnTrade, de forma predeterminada, la bandera vale false.

© 2000-2025, MetaQuotes Ltd.

OnTimerProcess

Establece la bandera para procesar el evento OnTimer.

```
void  OnTimerProcess(
   bool     value        // bandera
   )
```

Parámetros

value

[in] Bandera para procesar el evento OnTimer.

Valor devuelto

Ninguno.

Nota

Si la bandera es true, se procesa el evento OnTimer , de forma predeterminada, la bandera vale false.

© 2000-2025, MetaQuotes Ltd.

OnChartEventProcess

Establece la bandera para procesar el evento OnChartEvent.

```
void  OnChartEventProcess(
   bool     value        // bandera
   )
```

Parámetros

value

[in] Bandera para procesar el evento OnChartEvent.

Valor devuelto

Ninguno.

Nota

Si la bandera vale true, se procesa el evento OnChartEvent, de forma predeterminada, la bandera vale false.

© 2000-2025, MetaQuotes Ltd.

OnBookEventProcess

Establece la bandera que procesa el evento OnBookEvent.

```
void  OnChartEventProcess(
   bool     value        // bandera
   )
```

Parámetros

value

[in] Bandera que procesa el evento OnBookEvent.

Valor devuelto

Ninguno.

Nota

Si la bandera vale true, se procesa el evento OnBookEvent, de forma predeterminada, la bandera vale false.

© 2000-2025, MetaQuotes Ltd.

MaxOrders (Método Get)

Obtiene la cantidad máxima de órdenes permitidas.

```
int  MaxOrders()
```

Valor devuelto

Cantidad máxima de órdenes permitidas.

MaxOrders (Método Set)

Establece la cantidad máxima de órdenes permitidas.

```
void  MaxOrders(
   int      max_orders        // valor nuevo
   )
```

Parámetros

max_orders

[in] Valor nuevo de la cantidad máxima de órdenes permitidas.

Valor devuelto

Ninguno.

Nota

El número máximo de órdenes permitidas vale 1 de forma predeterminada.

© 2000-2025, MetaQuotes Ltd.

Signal

Obtiene el puntero al objeto Trade Signal.

```
CExpertSignal*  Signal() const
```

Valor devuelto

Puntero al objeto Trade Signal.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Comprueba las configuraciones.

```
virtual bool  ValidationSettings()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

También comprueba las configuraciones de todos los objetos de los Asesores Expertos.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

Inicializa todos los indicadores y series temporales.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

Las series temporales se inicializan si el objeto utiliza un símbolo o serie temporal diferentes a los definidos en la inicialización. Llama a los métodos virtuales InitIndicators() de la señal de trading, trailing stop y objetos de gestión del dinero.

© 2000-2025, MetaQuotes Ltd.

OnTick

Manejador del evento OnTick.

```
virtual void  OnTick()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

OnTrade

Manejador del evento OnTrade.

```
virtual void  OnTrade()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

OnTimer

Manejador del evento OnTimer.

```
virtual void  OnTimer()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

OnChartEvent

Manejador del evento OnChartEvent.

```
virtual void  OnChartEvent(
   const int     id,       // identificador del evento
   const long&   lparam,   // parámetro long
   const double  dparam,   // parámetro double
   const string  sparam    // parámetro string
   )
```

Parámetros

id

[in] ID del evento.

lparam

[in] Parámetro del evento de tipo long.

dparam

[in] Parámetro del evento de tipo double.

sparam

[in] Parámetro del evento de tipo string.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

OnBookEvent

Manejador del evento OnBookEvent.

```
virtual void  OnBookEvent(
   const string&     symbol       // símbolo
)
```

Parámetros

symbol

[in] Símbolo del evento OnBookEvent.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

InitParameters

Inicializa los parámetros del Asesor Experto.

```
virtual bool  InitParameters()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La función InitParameters() de la clase base CExpert no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

DeinitTrade

Desinicializa el objeto Trade.

```
virtual void  DeinitTrade()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

DeinitSignal

Desinicializa el objeto Signal.

```
virtual void  DeinitSignal()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

DeinitTrailing

Desinicializa el objeto Trailing.

```
virtual void  DeinitTrailing()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

DeinitMoney

Desinicializa el objeto de gestión del dinero.

```
virtual void  DeinitMoney()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

DeinitIndicators

Desinicializa todos los indicadores y todas las series temporales.

```
virtual void  DeinitIndicators()
```

Valor devuelto

Ninguno.

Nota

También desinicializa los indicadores y las series temporales de los objetos auxiliares.

© 2000-2025, MetaQuotes Ltd.

Refresh

Actualiza todos los datos.

```
virtual bool  Refresh()
```

Valor devuelto

true si hace falta procesar más ticks, de lo contrario false.

Nota

Permite determinar la necesidad de procesar los ticks. Si hace falta, actualiza todas las cotizaciones, las series temporales y los datos de los indicadores, y devuelve true.

Implementación

```
//+------------------------------------------------------------------+
//| Refresca los datos para procesarlos                              |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si se ejecuta correctamente, false en caso contrario.
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::Refresh()
  {
   MqlDateTime time;
//--- refresca los precios
   if(!m_symbol.RefreshRates()) return(false);
//--- comprueba si hace falta procesar
   TimeToStruct(m_symbol.Time(),time);
   if(m_period_flags!=WRONG_VALUE && m_period_flags!=0)
      if((m_period_flags & TimeframesFlags(time))==0) return(false);
   m_last_tick_time=time;
//--- refresca los indicadores
   m_indicators.Refresh();
//--- ok
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

Processing

Algoritmo de procesamiento principal.

```
virtual bool  Processing()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Lleva a cabo los siguientes pasos:

1. Comprueba la presencia en el símbolo de la posición abierta. Si no hay ninguna posición abierta, omite los pasos 2, 3 y 4.
2. Comprueba las condiciones para invertir la posición abierta (método CheckReverse()). Si la posición ha sido "revertida", termina.
3. Comprueba las condiciones para cerrar la posición (método CheckClose()). Si la posición se ha cerrado, omite el paso 4.
4. Comprueba las condiciones para modificar los parámetros de la posición (método CheckTrailingStop()). Si los parámetros de la posición han cambiado, termina.
5. Comprueba la presencia de órdenes pendientes en el símbolo. Si no hay ninguna orden pendiente, va al paso 9.
6. Comprueba las condiciones para borrar la orden (CheckDeleteOrderLong() para comprar órdenes pendientes o CheckDeleteOrderShort() para venderlas). Si la orden se ha borrado, va al paso 9.
7. Comprueba las condiciones para modificar los parámetros de la orden pendiente (CheckTrailingOrderLong() para comprar órdenes o CheckTrailingOrderShort() para venderlas). Si los parámetros de la orden han cambiado, termina.
8. Termina.
9. Comprueba las condiciones para abrir la posición (método CheckOpen()).

Si desea implementar su propio algoritmo, debe sobreescribir el método Processing() de la clase derivada.

Implementación

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Principal function                                               |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si se procesa alguna operación de trading, false en caso contrario.
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::Processing()
  {
//--- comprueba posiciones abiertas
   if(m_position.Select(m_symbol.Name()))
     {
      //--- abre posición disponible
      //--- comprueba la posibilidad de revertir la posición
      if(CheckReverse()) return(true);
      //--- comprueba la posibilidad de cerrar la posición/borrar órdenes pendientes
      if(!CheckClose())
        {
         //--- comprueba la posibilidad de modificar la posición
         if(CheckTrailingStop()) return(true);
         //--- devuelve sin operaciones
         return(false);
        }
     }
//--- comprueba si hay órdenes pendientes colocadas
   int total=OrdersTotal();
   if(total!=0)
     {
      for(int i=total-1;i>=0;i--)
        {
         m_order.SelectByIndex(i);
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         if(m_order.OrderType()==ORDER_TYPE_BUY_LIMIT || m_order.OrderType()==ORDER_TY
           {
            //--- comprueba la posibilidad de borrar una orden pendiente de compra
            if(CheckDeleteOrderLong()) return(true);
            //--- comprueba la posibilidad de modificar una orden pendiente de compra
            if(CheckTrailingOrderLong()) return(true);
           }
         else
           {
            //--- comprueba la posibilidad de borrar una orden pendiente de venta
            if(CheckDeleteOrderShort()) return(true);
            //--- comprueba la posibilidad de modificar una orden pendiente de venta
            if(CheckTrailingOrderShort()) return(true);
           }
         //--- devuelve sin operaciones
         return(false);
        }
     }
//--- comprueba la posibilidad de abrir una posición/establecer una orden pendiente
   if(CheckOpen()) return(true);
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

SelectPosition

Elegir una posición para trabajar posteriormente con ella.

```
void  SelectPosition()
```

Valor devuelto

No hay.

Puesta en práctica

```
//+------------------------------------------------------------------+
//| Position select                                                  |
//| INPUT:  no.                                                      |
//| OUTPUT: no.                                                      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::SelectPosition(void)
  {
   bool res=false;
//---
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      res=m_position.SelectByMagic(m_symbol.Name(),m_magic);
   else
      res=m_position.Select(m_symbol.Name());
//---
   return(res);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckOpen

Comprueba las condiciones de apertura de la posición.

```
virtual bool  CheckOpen()
```

Valor devuelto

true si se ha ejecutado alguna operación, false en caso contrario.

Nota

Comprueba las condiciones para abrir posiciones largas (CheckOpenLong()) y cortas (CheckOpenShort()).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba la apertura de la posición u orden limit/stop          |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckOpen()
  {
   if(CheckOpenLong())  return(true);
   if(CheckOpenShort()) return(true);
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Comprueba las condiciones de apertura de la posición larga.

```
virtual bool  CheckOpenLong()
```

Valor devuelto

true si se ha ejecutado alguna operación, false en caso contrario.

Nota

Comprueba las condiciones de apertura de la posición larga (método CheckOpenLong() del objeto Signal) y abre una posición larga (método OpenLong()) si es necesario.

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba la apertura de la posición larga u orden limit/stop    |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- comprueba la operación de entrada larga
   if(m_signal.CheckOpenLong(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenLong(price,sl,tp));
     }
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Comprueba las condiciones de apertura de la posición corta.

```
virtual bool  CheckOpenShort()
```

Valor devuelto

true si se ha ejecutado alguna operación, false en caso contrario.

Nota

Comprueba las condiciones de apertura de la posición corta (método CheckOpenShort() del objeto Signal) y abre una posición corta (método OpenShort()) si es necesario.

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba la apertura de la posición corta u orden limit/stop    |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- comprueba la señal para las operaciones de entrada cortas
   if(m_signal.CheckOpenShort(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenShort(price,sl,tp));
     }
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

OpenLong

Abre una posición larga.

```
virtual bool  OpenLong(
   double    price,    // precio
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

tp

[in] Precio Take Profit.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Obtiene el volumen (método LotOpenLong(...)) y abre una posición larga (método Buy() del objeto Trade) si el volumen de trading no es igual a 0.

Implementación

```
//+------------------------------------------------------------------+
//| Abre posición larga o establece orden limit/stop                 |
//| ENTRADA:  price - precio,                                        |
//|           sl    - stop loss,                                     |
//|           tp    - take profit.                                   |
//| SALIDA: true si se procesa la operación, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::OpenLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- obtiene el lote para abrir
   double lot=LotOpenLong(price,sl);
//--- comprueba el lote para abrir
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

OpenShort

Abre una posición corta.

```
virtual bool  OpenShort(
   double    price,    // precio
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

tp

[in] Precio Take Profit.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Obtiene el volumen (método LotOpenShort()) y abre una posición corta (llamando al método Sell del objeto Trade) si el volumen de trading no es igual a 0.

Implementación

```
//+------------------------------------------------------------------+
//| Abre posición corta o establece orden limit/stop                 |
//| ENTRADA:  price - precio,                                        |
//|           sl    - stop loss,                                     |
//|           tp    - take profit.                                   |
//| SALIDA: true si la operación se ejecuta correctamente, false en caso contrario.
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::OpenShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- obtiene lote para abrir
   double lot=LotOpenShort(price,sl);
//--- comprueba lote para abrir
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverse

Comprueba las condiciones para revertir la posición abierta.

```
virtual bool  CheckReverse()
```

Valor devuelto

true si se ha ejecutado alguna operación, false en caso contrario.

Nota

Comprueba las condiciones para revertir las posiciones largas (CheckReverseLong()) y cortas (CheckReverseShort()).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba la reversión de la posición                            |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckReverse()
  {
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- comprueba la posibilidad de revertir la posición larga
      if(CheckReverseLong())  return(true);
     }
   else
      //---comprueba la posibilidad de revertir la posición corta
      if(CheckReverseShort()) return(true);
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverseLong

Comprueba las condiciones para revertir la posición larga.

```
virtual bool  CheckReverseLong()
```

Valor devuelto

true si se ha ejecutado alguna operación, false en caso contrario.

Nota

Comprueba las condiciones para revertir la posición larga (método CheckReverseLong() del objeto Signal) y realiza la operación inversa de la posición larga actual (método ReverseLong(...)) si es necesario.

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba la reversión de la posición larga                      |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- comprueba la señal para revertir operaciones largas
   if(m_signal.CheckReverseLong(price,sl,tp,expiration)) return(ReverseLong(price,sl,t
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverseShort

Comprueba las condiciones para revertir la posición corta.

```
virtual bool  CheckReverseLong()
```

Valor devuelto

true si se ha ejecutado alguna operación, false en caso contrario.

Nota

Comprueba las condiciones para revertir la posición corta (método CheckReverseShort() del objeto Signal) y realiza la operación inversa de la posición corta actual (método ReverseShort()) si es necesario.

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba la reversión de la posición corta                      |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- comprueba la señal para revertir operaciones cortas
   if(m_signal.CheckReverseShort(price,sl,tp,expiration)) return(ReverseShort(price,sl
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

ReverseLong

Realiza la operación inversa de la posición larga.

```
virtual bool  ReverseLong(
   double    price,    // precio
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

tp

[in] Precio Take Profit.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Obtiene el volumen de la posición inversa (método LotReverse()) y realiza la operación de la posición larga inversa (método Sell() del objeto Trade) si el volumen de trading no es igual a 0.

Implementación

```
//+------------------------------------------------------------------+
//| Posición larga inversa                                           |
//| ENTRADA:  price - precio,                                        |
//|           sl    - stop loss,                                     |
//|           tp    - take profit.                                   |
//| SALIDA: true si se procesa la operación, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::ReverseLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- obtiene el lote de la posición inversa
   double lot=LotReverse(sl);
//--- comprueba el lote
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

ReverseShort

Realiza la operación inversa de la posición corta.

```
virtual bool  ReverseShort(
   double    price,    // precio
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

tp

[in] Precio Take Profit.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Obtiene el volumen de la posición inversa (método LotReverse(...)) y realiza la operación de trading de la posición corta inversa (método Buy() del objeto Trade) si el volumen de trading no es igual a 0.

Implementación

```
//+------------------------------------------------------------------+
//| Posición corta inversa                                           |
//| ENTRADA:  price - precio,                                        |
//|           sl    - stop loss,                                     |
//|           tp    - take profit.                                   |
//| SALIDA: true si se procesa la operación, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::ReverseShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- obtiene el lote de la posición inversa
   double lot=LotReverse(sl);
//--- comprueba el lote
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckClose

Comprueba las condiciones de cierre de la posición.

```
virtual bool  CheckClose()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

1. Comprueba las condiciones Stop Out del Asesor Experto (el método CheckClose() del objeto de gestión del dinero). Si la condición se satisface, cierra la posición, borra todas las órdenes (CloseAll(...)) y sale.
2. Comprueba las condiciones de cierre de la posición larga o corta (métodos CheckCloseLong() o CheckCloseShort()) y si la posición se cierra, borra todas las órdenes (método DeleteOrders()).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba el cierre de la posición o borrado de la orden limit/stop
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.      |
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckClose()
  {
   double lot;
//--- la posición debe seleccionarse antes de la llamada
   if((lot=m_money.CheckClose(GetPointer(m_position)))!=0.0)
      return(CloseAll(lot));
//--- comprueba el tipo de posición
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- comprueba la posibilidad de cerrar la posición larga / borrar las posicior
      if(CheckCloseLong())
        {
         DeleteOrders();
         return(true);
        }
     }
   else
     {
      //--- comprueba la posibilidad de cerrar la posición corta / borrar las posicior
      if(CheckCloseShort())
        {
         DeleteOrders();
         return(true);
        }
     }
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckCloseLong

Comprueba las condiciones de cierre de la posición larga.

```
virtual bool  CheckCloseLong()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Comprueba las condiciones de cierre de la posición larga (método CheckCloseLong() del objeto Signal) y si se satisfacen, cierra la posición abierta (método CloseLong(...)).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba el cierre de la posición larga o borrado de la orden limit/stop
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.      |
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseLong()
  {
   double price=EMPTY_VALUE;
//--- comprueba el cierre de las operaciones largas
   if(m_signal.CheckCloseLong(price))
      return(CloseLong(price));
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckCloseShort

Comprueba las condiciones de cierre de la posición corta.

```
virtual bool  CheckCloseShort()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Comprueba las condiciones de cierre de la posición corta (método CheckCloseShort() del objeto Signal) y si se satisfacen, cierra la posición (método CloseShort()).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba el cierre de la posición corta o borrado de la orden limit/stop
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.      |
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseShort()
  {
   double price=EMPTY_VALUE;
//--- comprueba el cierre de las posiciones cortas
   if(m_signal.CheckCloseShort(price))
      return(CloseShort(price));
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CloseAll

Realiza el cierre parcial de las posiciones.

```
virtual bool  CloseAll(
   double    lot    // lote
   )
```

Parámetros

lot

[in] Número de lotes con que reducir la posición.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

En el modo de "compensación" del registro de posiciones, el cierre se produce con los métodos CExpertTrade::Buy o CExpertTrade::Sell. En el modo de "cobertura", con el método CTrade::PositionClose, que también se puede usar en las cuentas con el sistema de compensación. Para eliminar todas las órdenes pendientes, se utiliza el método DeleteOrders().

Implementación

```
//+------------------------------------------------------------------+
//| Cierre de la posición y borrado de órdenes                       |
//| ENTRADA:  lote - volumen de cierre.                              |
//| SALIDA: true si la operación se procesa; false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CloseAll(double lot)
  {
   bool result;
//--- comprueba el cierre de las operaciones
   if(m_position.PositionType()==POSITION_TYPE_BUY) result=m_trade.Sell(lot,0,0,0);
   else                                             result=m_trade.Buy(lot,0,0,0);
   result|=DeleteOrders();
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

Close

Cierra la posición abierta.

```
virtual bool  Close()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Cierra la posición (método PositionClose() del objeto de la clase CTrade).

Implementación

```
//+------------------------------------------------------------------+
//| Cierra la posición                                               |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::Close()
  {
   return(m_trade.PositionClose(m_symbol.Name()));
  }
```

© 2000-2025, MetaQuotes Ltd.

CloseLong

Cierra la posición larga.

```
virtual bool  CloseLong(
   double    price    // precio
   )
```

Parámetros

price

[in] Precio.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Cierra la posición larga (método Sell(...) del objeto de la clase CTrade).

Implementación

```
//+------------------------------------------------------------------+
//| Cierre de la posición larga                                      |
//| ENTRADA:  price - precio de cierre.                              |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CloseLong(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Sell(m_position.Volume(),price,0,0));
  }
```

© 2000-2025, MetaQuotes Ltd.

CloseShort

Cierra la posición corta.

```
virtual bool  CloseShort(
   double    price    // precio
   )
```

Parámetros

price

[in] Precio.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Cierra la posición corta (método Buy(...) del objeto de la clase CTrade).

Implementación

```
//+------------------------------------------------------------------+
//| Cierre de la posición corta                                      |
//| ENTRADA:  price - precio de cierre.                              |
//| SALIDA: true si la operación se ejecuta correctamente, false en caso contrario.
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CloseShort(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Buy(m_position.Volume(),price,0,0));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStop

Comprueba las condiciones Trailing Stop de la posición abierta.

```
virtual bool  CheckTrailingStop()
```

Valor devuelto

true si se ha ejecutado alguna operación, false en caso contrario.

Nota

Comprueba las condiciones Trailing Stop de la posición abierta (CheckTrailingStopLong() o CheckTrailingStopShort() para las posiciones largas y cortas).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba el trailing stop/profit de la posición                 |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStop()
  {
//--- la posición se tiene que seleccionar antes de la llamada
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- comprueba la posibilidad de modificar la posición larga
      if(CheckTrailingStopLong()) return(true);
     }
   else
     {
      //--- comprueba la posibilidad de modificar la posición corta
      if(CheckTrailingStopShort()) return(true);
     }
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Comprueba las condiciones Trailing Stop de la posición larga abierta.

```
virtual bool  CheckTrailingStopLong()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Comprueba las condiciones Trailing Stop de la posición larga abierta (método CheckTrailingStopLong(...) del objeto Expert Trailing). Si las condiciones se satisfacen, modifica los parámetros de la posición (método TrailingStopLong(...)).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba el trailing stop/profit de la posición larga           |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopLong()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- comprueba el trailing stop de las operaciones largas
   if(m_trailing.CheckTrailingStopLong(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- operaciones trailing stop largas
      return(TrailingStopLong(sl,tp));
     }
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Comprueba las condiciones Trailing Stop de la posición corta abierta.

```
virtual bool  CheckTrailingStopShort()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Comprueba las condiciones Trailing Stop de la posición corta abierta (método CheckTrailingStopShort(...) del objeto Expert Trailing). Si las condiciones se satisfacen, modifica los parámetros de la posición (método TrailingStopShort(...)).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba el trailing stop/profit de la posición corta           |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopShort()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- comprueba las operaciones trailing stop cortas
   if(m_trailing.CheckTrailingStopShort(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- operaciones cortas trailing stop
      return(TrailingStopShort(sl,tp));
     }
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingStopLong

Modifica los parámetros de la posición larga abierta.

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Parámetros

sl

[in] Precio Stop Loss.

tp

[in] Precio Take Profit.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

La función modifica los parámetros de la posición larga abierta (método PositionModify(...) del objeto de la clase CTrade).

Implementación

```
//+------------------------------------------------------------------+
//| Trailing de la posición larga stop/profit                        |
//| ENTRADA:  sl - nuevo stop loss,                                  |
//|           tp - nuevo take profit.                                |
//| SALIDA: true si la operación tiene éxito, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopLong(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingStopShort

Modifica los parámetros de la posición corta abierta.

```
virtual bool  TrailingStopLong(
   double    sl,     // Stop Loss
   double    tp,     // Take Profit
   )
```

Parámetros

sl

[in] Precio Stop Loss.

tp

[in] Precio Take Profit.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

La función modifica los parámetros de la posición corta abierta (método PositionModify(...) del objeto de la clase CTrade).

Implementación

```
//+------------------------------------------------------------------+
//| Trailing de la posición corta stop/profit                        |
//| ENTRADA:  sl - nuevo stop loss,                                  |
//|           tp - nuevo take profit.                                |
//| SALIDA: true si la operación tiene éxito, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopShort(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderLong

Comprueba las condiciones Trailing Stop de la orden pendiente Buy Limit/Stop.

```
virtual bool  CheckTrailingOrderLong()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Comprueba las condiciones Trailing Stop de la orden pendiente Buy Limit/Stop (método CheckTrailingOrderLong() del objeto Trade Signals) y modifica los parámetros de la orden si es necesario (método TrailingOrderLong(...)).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba el trailing de la orden larga limit/stop               |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderLong()
  {
   double price;
//--- comprueba la posibilidad de modificar la orden larga
   if(m_signal.CheckTrailingOrderLong(GetPointer(m_order),price))
      return(TrailingOrderLong(m_order.PriceOpen()-price));
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderShort

Comprueba las condiciones Trailing Stop de la orden pendiente Sell Limit/Stop.

```
virtual bool  CheckTrailingOrderShort()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Comprueba las condiciones Trailing Stop de la orden pendiente Sell Limit/Stop (método CheckTrailingOrderShort() del objeto Trade Signals) y modifica los parámetros de la orden si es necesario (método TrailingOrderShort()).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba el trailing de la orden corta limit/stop               |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderShort()
  {
   double price;
//--- comprueba la posibilidad de modificar la orden corta
   if(m_signal.CheckTrailingOrderShort(GetPointer(m_order),price))
      return(TrailingOrderShort(m_order.PriceOpen()-price));
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingOrderLong

Modifica los parámetros de la orden pendiente Buy Limit/Stop.

```
virtual bool  TrailingOrderLong(
   double    delta    // delta
   )
```

Parámetros

delta

[in] Precio delta.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Modifica los parámetros de la orden pendiente Buy Limit/Stop (método OrderModify(...) del objeto de la clase CTrade).

Implementación

```
//+------------------------------------------------------------------+
//| Trailing de la orden larga limit/stop                            |
//| ENTRADA:  delta - cambio de precio.                              |
//| SALIDA: true si la operación es exitosa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderLong(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modifica la orden larga
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingOrderShort

Modifica los parámetros de la orden pendiente Sell Limit/Stop.

```
virtual bool  TrailingOrderShort(
   double    delta    // delta
   )
```

Parámetros

delta

[in] Precio delta.

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Modifica los parámetros de la orden pendiente Sell Limit/Stop (método OrderModify(...) del objeto de la clase CTrade).

Implementación

```
//+------------------------------------------------------------------+
//| Trailing de la orden corta limit/stop                            |
//| INPUT:  delta - price change.                                    |
//| SALIDA: true si la operación es exitosa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderShort(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modifica la orden corta
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckDeleteOrderLong

Comprueba las condiciones para borrar la orden pendiente Buy Limit/Stop.

```
virtual bool  CheckDeleteOrderLong()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Comprueba la fecha de caducidad de la orden. Comprueba las condiciones para borrar la orden pendiente Sell Limit/Stop (método CheckCloseLong(...) del objeto de la clase Signal) y borra la orden si la condición se satisface (método DeleteOrderLong()).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba el borrado de la orden larga limit/stop                |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderLong()
  {
   double price;
//--- comprueba la posibilidad de borrar la orden larga
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderLong());
     }
   if(m_signal.CheckCloseLong(price))
      return(DeleteOrderLong());
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckDeleteOrderShort

Comprueba las condiciones para borrar la orden pendiente Sell Limit/Stop.

```
virtual bool  CheckDeleteOrderShort()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Comprueba la fecha de caducidad de la orden. Comprueba las condiciones para borrar la orden pendiente Sell Limit/Stop (método CheckCloseShort(...) del objeto de la clase Signal) y borra la orden si la condición se satisface (método DeleteOrderShort()).

Implementación

```
//+------------------------------------------------------------------+
//| Comprueba el borrado de la orden corta limit/stop                |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se procesa, false en caso contrario.|
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderShort()
  {
   double price;
//--- comprueba la posibilidad de borrar la orden corta
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderShort());
     }
   if(m_signal.CheckCloseShort(price))
      return(DeleteOrderShort());
//--- devuelve sin operaciones
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrders

Borra todas las órdenes.

```
virtual bool  DeleteOrders()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Borra todas las órdenes (DeleteOrder() para todas las órdenes).

Implementación

```
//+------------------------------------------------------------------+
//| Borra todas las órdenes limit/stop                               |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se ejecuta correctamente, false en caso contrario.
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrders()
  {
   bool result=false;
   int  total=OrdersTotal();
//---
   for(int i=total-1;i>=0;i--)
     {
      if(m_order.Select(OrderGetTicket(i)))
        {
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         result|=DeleteOrder();
        }
     }
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrder

Borra la orden pendiente Limit/Stop.

```
virtual bool  DeleteOrder()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Borra la orden pendiente Limit/Stop (método OrderDelete(...) del objeto de la clase CTrade).

Implementación

```
//+------------------------------------------------------------------+
//| Borra la orden limit/stop                                        |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se ejecuta correctamente, false en caso contrario.
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrder()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrderLong

Borra la orden pendiente Buy Limit/Stop.

```
virtual bool  DeleteOrderLong()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Borra la orden pendiente Buy Limit/Stop (método OrderDelete(...) del objeto de la clase CTrade).

Implementación

```
//+------------------------------------------------------------------+
//| Borra la orden larga limit/stop                                  |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se ejecuta correctamente, false en caso contrario.
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderLong()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrderShort

Borra la orden pendiente Sell Limit/Stop.

```
virtual bool  DeleteOrderShort()
```

Valor devuelto

true si la operación se ha ejecutado; en caso contrario false.

Nota

Borra la orden pendiente Sell Limit/Stop (método OrderDelete(...) del objeto de la clase CTrade).

Implementación

```
//+------------------------------------------------------------------+
//| Borra la orden corta limit/stop                                  |
//| ENTRADA:  no.                                                    |
//| SALIDA: true si la operación se ejecuta correctamente, false en caso contrario.
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderShort()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotOpenLong

Obtiene el volumen de la operación de compra.

```
double  LotOpenLong(
   double    price,    // precio
   double    sl        // Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen (en lotes) de la operación de compra.

Nota

Obtiene el volumen de la operación de compra (método CheckOpenLong(...) del objeto de gestión del dinero).

Implementación

```
//+------------------------------------------------------------------+
//| Método que obtiene el lote de la posición larga abierta.         |
//| ENTRADA:  price - precio,                                        |
//|           sl    - stop loss.                                     |
//| SALIDA: lote de apertura.                                        |
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
double CExpert::LotOpenLong(double price,double sl)
  {
   return(m_money.CheckOpenLong(price,sl));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotOpenShort

Obtiene el volumen de la operación de venta.

```
double  LotOpenShort(
   double    price,   // precio
   double    sl       // Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen (en lotes) de la operación de venta.

Nota

Obtiene el volumen de la operación de venta (método CheckOpenShort(...) del objeto de gestión del dinero).

Implementación

```
//+------------------------------------------------------------------+
//| Método que obtiene el lote de la posición corta abierta.         |
//| ENTRADA:  price - precio,                                        |
//|           sl    - stop loss.                                     |
//| SALIDA: lote de apertura.                                        |
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
double CExpert::LotOpenShort(double price,double sl)
  {
   return(m_money.CheckOpenShort(price,sl));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotReverse

Obtiene el volumen de la posición opuesta.

```
double  LotReverse(
   double    sl       // Stop Loss
   )
```

Parámetros

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen (en lotes) de la operación de la posición opuesta.

Nota

Obtiene el volumen de la operación de la posición opuesta (método CheckReverse(...) del objeto de gestión del dinero).

Implementación

```
//+------------------------------------------------------------------+
//| Método que obtiene el lote de la posición opuesta.               |
//| ENTRADA:  sl - stop loss.                                        |
//| SALIDA: lote de apertura.                                        |
//| OBSERVACIONES: no.                                               |
//+------------------------------------------------------------------+
double CExpert::LotReverse(double sl)
  {
   return(m_money.CheckReverse(GetPointer(m_position),sl));
  }
```

© 2000-2025, MetaQuotes Ltd.

PrepareHistoryDate

Establece la fecha de inicio del seguimiento del historial de trading.

```
void  PrepareHistoryDate()
```

Nota

El período de seguimiento del historial de trading se establece desde el principio del mes (pero no menos de un día).

© 2000-2025, MetaQuotes Ltd.

HistoryPoint

Crea un punto de control del historial de trading (guarda el número de posiciones, órdenes, transacciones e historial de órdenes).

```
void  HistoryPoint(
   bool   from_check_trade=false   // bandera
   )
```

Parámetros

from_check_trade=false

[in] Bandera para evitar la recursión.

Nota

Guarda la cantidad de posiciones, órdenes, transacciones e historial de órdenes.

© 2000-2025, MetaQuotes Ltd.

CheckTradeState

Compara el estado actual con el guardado, y llama al manejador de evento correspondiente.

```
bool  CheckTradeState()
```

Valor devuelto

true si se ha manejado el evento; false en caso contrario.

Nota

Comprueba el número de posiciones, órdenes, transacciones y órdenes históricas comparando los valores guardados con el método HistoryPoint() . Si el historial de trading ha cambiado llama al manejador de evento correspondiente.

© 2000-2025, MetaQuotes Ltd.

WaitEvent

Establece la bandera de espera del evento.

```
void  WaitEvent(
   ENUM_TRADE_EVENTS    event          // bandera
   )
```

Parámetros

event

[in] Bandera con los eventos a establecer (enumeración ENUM_TRADE_EVENTS).

Valor devuelto

Ninguno.

Banderas de eventos

```
//--- banderas de espera de eventos
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT               =0,         // sin esperar eventos
   TRADE_EVENT_POSITION_OPEN          =0x1,       // bandera que espera el evento "aper
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,        // bandera que espera el evento "modi
   TRADE_EVENT_POSITION_MODIFY        =0x4,       // bandera que espera el evento "modi
   TRADE_EVENT_POSITION_CLOSE         =0x8,       // bandera que espera el evento "cier
   TRADE_EVENT_POSITION_STOP_TAKE     =0x10,      // bandera que espera el evento "disp
   TRADE_EVENT_ORDER_PLACE            =0x20,      // bandera que espera el evento "colo
   TRADE_EVENT_ORDER_MODIFY           =0x40,      // bandera que espera el evento "modi
   TRADE_EVENT_ORDER_DELETE           =0x80,      // bandera que espera el evento "elim
   TRADE_EVENT_ORDER_TRIGGER          =0x100      // bandera que espera el evento "disp
  };
```

© 2000-2025, MetaQuotes Ltd.

NoWaitEvent

Reinicia la bandera de espera del evento.

```
void  NoWaitEvent(
   ENUM_TRADE_EVENTS    event         // bandera
   )
```

Parámetros

event

[in] Bandera con los eventos a reiniciar (enumeración ENUM_TRADE_EVENTS).

Valor devuelto

Ninguno.

Banderas de eventos

```
//--- banderas de espera de eventos
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT               =0,         // sin esperar eventos
   TRADE_EVENT_POSITION_OPEN          =0x1,       // bandera que espera el evento "aper
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,        // bandera que espera el evento "modi
   TRADE_EVENT_POSITION_MODIFY        =0x4,       // bandera que espera el evento "modi
   TRADE_EVENT_POSITION_CLOSE         =0x8,       // bandera que espera el evento "cier
   TRADE_EVENT_POSITION_STOP_TAKE     =0x10,      // bandera que espera el evento "disp
   TRADE_EVENT_ORDER_PLACE            =0x20,      // bandera que espera el evento "colo
   TRADE_EVENT_ORDER_MODIFY           =0x40,      // bandera que espera el evento "modi
   TRADE_EVENT_ORDER_DELETE           =0x80,      // bandera que espera el evento "elim
   TRADE_EVENT_ORDER_TRIGGER          =0x100      // bandera que espera el evento "disp
  };
```

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionStopTake

Manejador del evento "Posición Stop Loss/Take Profit disparada".

```
virtual bool  TradeEventPositionStopTake()
```

Valor devuelto

El método de la clase CExpert no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderTriggered

Manejador del evento "Orden pendiente disparada".

```
virtual bool  TradeEventOrderTriggered()
```

Valor devuelto

El método de la clase CExpert no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionOpened

Manejador del evento "Posición abierta".

```
virtual bool  TradeEventPositionOpened()
```

Valor devuelto

El método de la clase CExpert no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionVolumeChanged

Manejador del evento "Volumen de la posición modificado".

```
virtual bool  TradeEventPositionVolumeChanged()
```

Valor devuelto

El método de la clase CExpert no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionModified

Manejador del evento "Posición modificada".

```
virtual bool  TradeEventPositionModified()
```

Valor devuelto

El método de la clase CExpert no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionClosed

Manejador del evento "Posición cerrada".

```
virtual bool  TradeEventPositionClosed()
```

Valor devuelto

El método de la clase CExpert no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderPlaced

Manejador del evento "Orden pendiente colocada".

```
virtual bool  TradeEventOrderPlaced()
```

Valor devuelto

El método de la clase CExpert no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderModified

Manejador del evento "Orden pendiente modificada".

```
virtual bool  TradeEventOrderModified()
```

Valor devuelto

El método de la clase CExpert no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderDeleted

Manejador del evento "Orden pendiente eliminada".

```
virtual bool  TradeEventOrderDeleted()
```

Valor devuelto

El método de la clase CExpert no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

TradeEventNotIdentified

Manejador del evento no identificado.

```
virtual bool  TradeEventNotIdentified()
```

Valor devuelto

El método de la clase CExpert no hace nada y siempre devuelve true.

Nota

Cabe señalar que pueden llegar varios eventos de trading, en cuyo caso es difícil identificarlos.

© 2000-2025, MetaQuotes Ltd.

TimeframeAdd

Añade el periodo de tiempo para realizar el seguimiento.

```
void  TimeframeAdd(
   ENUM_TIMEFRAMES    period         // periodo temporal
   )
```

Parámetros

period

[in] Periodo temporal (enumeración ENUM_TIMEFRAMES).

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

TimeframesFlags

El método devuelve la bandera que indica los periodos temporales con una nueva barra.

```
int  TimeframesFlags(
   MqlDateTime&    time         // variable para la fecha y la hora
   )
```

Parámetros

time

[in] Variable de tipo MqlDateTime para la nueva hora, pasada por referencia.

Valor devuelto

Devuelve la bandera que indica los periodos temporales con una nueva barra.

© 2000-2025, MetaQuotes Ltd.

CExpertSignal

CExpertSignal es una clase base para las señales de trading, no hace nada (a excepción de los métodos CheckReverseLong() y CheckReverseShort()) salvo proporcionar las interfaces.

Cómo utilizarla:

1. Prepare un algoritmo para las señales de trading;
2. Cree su propia clase de trading, haciéndola heredar de CExpertSignal;
3. Sobreescriba los métodos virtuales de su clase, escribiendo sus propios algoritmos.

Puede encontrar algunos ejemplos de clases de señales de trading en la carpeta Expert\Signal\.

Descripción

CExpertSignal es una clase base para la implementación de algoritmos de señales de trading.

Declaración

```
class CExpertSignal : public CExpertBase
```

Título

```
#include <Expert\ExpertSignal.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpertSignal

Descendientes directos

CSignalAC, CSignalAMA, CSignalAO, CSignalBearsPower, CSignalBullsPower, CSignalCCI, CSignalDeM, CSignalDEMA, CSignalEnvelopes, CSignalFrAMA, CSignalRSI, CSignalRVI, CSignalSAR, CSignalStoch, CSignalTEMA, CSignalTriX, CSignalWPR

Métodos de la clase

Inicialización	
virtual InitIndicators	Inicializa los indicadores y las series temporales
virtual ValidationSettings	Comprueba las configuraciones
virtual AddFilter	Añade un filtro a una señal combinada
Acceso a datos protegidos	
BasePrice	Establece el nivel del precio base
UsedSeries	Obtiene las banderas de la serie temporal utilizada
Establecimiento de los parámetros	

© 2000-2025, MetaQuotes Ltd.

Inicialización	
Weight	Establece el valor del parámetro "Peso"
PatternsUsage	Establece el valor del parámetro "Uso de patrones"
General	Establece el valor del parámetro "General"
Ignore	Establece el valor del parámetro "Ignorar"
Invert	Establece el valor del parámetro "Invertir"
ThresholdOpen	Establece el valor del parámetro "Abrir umbral"
ThresholdClose	Establece el valor del parámetro "Cerrar umbral"
PriceLevel	Establece el valor del parámetro "Nivel de precio"
StopLevel	Establece el valor del parámetro "Nivel stop"
TakeLevel	Establece el valor del parámetro "Nivel take"
Expiration	Establece el valor del parámetro "Vencimiento"
Magic	Establece el valor del parámetro "Magia"
Comprobación de condiciones	
virtual CheckOpenLong	Comprueba las condiciones de apertura de la posición larga
virtual CheckCloseLong	Comprueba las condiciones de cierre de la posición larga
virtual CheckOpenShort	Comprueba las condiciones de apertura de la posición corta
virtual CheckCloseShort	Comprueba las condiciones de cierre de la posición corta
virtual CheckReverseLong	Comprueba las condiciones de reversión de la posición larga
virtual CheckReverseShort	Comprueba las condiciones de reversión de la posición corta
Establecimiento de parámetros	
virtual OpenLongParams	Establece los parámetros de apertura de la posición larga
virtual OpenShortParams	Establece los parámetros de apertura de la posición corta
virtual CloseLongParams	Establece los parámetros de cierre de la posición larga
virtual CloseShortParams	Establece los parámetros de cierre de la posición corta
Comprueba las condiciones de Order Trailing	
virtual CheckTrailingOrderLong	Comprueba las condiciones para modificar los parámetros de la orden Buy Pending
virtual CheckTrailingOrderShort	Comprueba las condiciones para modificar los parámetros de la orden Sell Pending

© 2000-2025, MetaQuotes Ltd.

Inicialización	
Métodos para comprobar la formación de órdenes de mercado	
virtual LongCondition	Obtiene el resultado de la comprobación de las condiciones de compra
virtual ShortCondition	Obtiene el resultado de la comprobación de las condiciones de venta
virtual Direction	Obtiene la dirección "ponderada" del precio

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

BasePrice

Establece el nivel del precio base.

```
void  BasePrice(
   double    value         // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo del nivel del precio base.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

UsedSeries

Obtiene las banderas de las series temporales utilizadas.

```
int  UsedSeries()
```

Valor devuelto

Banderas de las series temporales utilizadas (si el símbolo/serie temporal se corresponde con el símbolo/serie temporal actual), en caso contrario 0.

© 2000-2025, MetaQuotes Ltd.

Weight

Establece el nuevo valor del parámetro "Peso".

```
void  Weight(
   double    value          // valor nuevo
   )
```

Parámetros

value

[in] Nuevo valor del parámetro "Peso".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

PatternUsage

Establece el nuevo valor del parámetro "Usar patrones".

```
void  PatternUsage(
   double    value          // valor nuevo
   )
```

Parámetros

value

[in] Nuevo valor de "Usar patrones".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

General

Establece el valor nuevo del parámetro "General".

```
void  General(
   int     value          // valor nuevo
   )
```

Parámetros

value

[in] Nuevo valor de "General".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Ignore

Establece el nuevo valor del parámetro "Ignorar".

```
void  Ignore(
   long     value          // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo de "Ignorar".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Invert

Establece el nuevo valor del parámetro "Invertir".

```
void  Invert(
   long     value          // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo de "Invertir".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

ThresholdOpen

Establece el nuevo valor del parámetro "Umbral de apertura".

```
void  ThresholdOpen(
   long     value          // valor nuevo
   )
```

Parámetros

value

[in] Nuevo valor del "Umbral de apertura".

Valor devuelto

Ninguno.

Nota

El rango del parámetro "Umbral de apertura" está comprendido entre 0 y 100. Utilizado cuando se "vota" por abrir la posición.

© 2000-2025, MetaQuotes Ltd.

ThresholdClose

Establece el nuevo valor del parámetro "Umbral de cierre".

```
void  ThresholdOpen(
   long     value          // valor nuevo
   )
```

Parámetros

value

[in] Nuevo valor del "Umbral de cierre".

Valor devuelto

Ninguno.

Nota

El rango del parámetro "Umbral de cierre" está comprendido entre 0 y 100. Utilizado cuando se "vota" por cerrar la posición.

© 2000-2025, MetaQuotes Ltd.

PriceLevel

Establece el nuevo valor del parámetro "Nivel del precio".

```
void  PriceLevel(
   double    value          // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo del "Nivel del precio".

Valor devuelto

Ninguno.

Nota

El valor del "Nivel del precio" se define en unidades de nivel de precio. El método PriceLevelUnit() devuelve los valores numéricos de las unidades del nivel del precio. El "Nivel del precio" se utiliza para definir el precio de apertura en relación al precio base.

© 2000-2025, MetaQuotes Ltd.

StopLevel

Establece el nuevo valor del parámetro "Nivel stop".

```
void  StopLevel(
   double    value         // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo de "Nivel stop".

Valor devuelto

Ninguno.

Nota

El valor de "Nivel stop" se define en unidades de nivel de precio. El método PriceLevelUnit() devuelve el valor numérico de la unidad del nivel de precio. El "Nivel stop" se utiliza para definir el precio Stop Loss relativo al precio de apertura.

© 2000-2025, MetaQuotes Ltd.

TakeLevel

Establece el nuevo valor del parámetro "Nivel take".

```
void  TakeLevel(
   double    value          // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo del "Nivel take".

Valor devuelto

Ninguno.

Nota

El valor de "Nivel take" se define en unidades de nivel de precio. El método PriceLevelUnit() devuelve el valor numérico de la unidad del nivel de precio. El "Nivel take" se utiliza para definir el precio Take Profit relativo al precio de apertura.

© 2000-2025, MetaQuotes Ltd.

Expiration

Establece el valor del parámetro "Vencimiento".

```
void  Expiration(
   int     value          // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo del "Vencimiento".

Valor devuelto

Ninguno.

Nota

El valor del parámetro "Vencimiento" se define en las barras. Se utiliza como hora de Vencimiento en las órdenes pendientes (cuando se hace trading con órdenes pendientes).

© 2000-2025, MetaQuotes Ltd.

Magic

Establece el valor del parámetro "Magia".

```
void  Magic(
   int     value          // valor nuevo
   )
```

Parámetros

value

[in] Nuevo valor de la "Magia" (ID del Asesor Experto).

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Comprueba las configuraciones.

```
virtual bool  ValidationSettings()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

Inicializa todos los indicadores y series temporales.

```
virtual bool  InitIndicators(
   CIndicators*  indicators    // puntero
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La serie temporal solo se inicializa si el objeto utiliza un símbolo o periodo temporal diferentes a los definidos en la inicialización.

© 2000-2025, MetaQuotes Ltd.

AddFilter

Añade un filtro a la señal compuesta.

```
virtual bool  AddFilter(
   CExpertSignal*  filter    // puntero
   )
```

Parámetros

indicators

[in] Puntero al objeto filtro.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Comprueba las condiciones de apertura de la posición larga.

```
virtual bool  CheckOpenLong(
   double&    price,          // precio
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // vencimiento
   )
```

Parámetros

price

[in][out] Variable del precio, pasada por referencia.

sl

[in][out] Variable del precio Stop Loss, pasada por referencia.

tp

[in][out] Variable del precio Take Profit, pasada por referencia.

expiration

[in][out] Variable de la hora de vencimiento, pasada por referencia.

Valor devuelto

true si la condición se satisface, false en otro caso.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Comprueba las condiciones de apertura de la posición corta.

```
virtual bool  CheckOpenShort(
   double&    price,          // precio
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // vencimiento
   )
```

Parámetros

price

[in][out] Variable del precio, pasada por referencia.

sl

[in][out] Variable del precio Stop Loss, pasada por referencia.

tp

[in][out] Variable del precio Take Profit, pasada por referencia.

expiration

[in][out] Variable de la hora de vencimiento, pasada por referencia.

Valor devuelto

true si la condición se satisface, false en otro caso.

© 2000-2025, MetaQuotes Ltd.

OpenLongParams

Establece los parámetros de apertura de la posición larga.

```
virtual bool  OpenLongParams(
   double&    price,          // precio
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // vencimiento
   )
```

Parámetros

price

[in][out] Variable del precio, pasada por referencia.

sl

[in][out] Variable del precio Stop Loss, pasada por referencia.

tp

[in][out] Variable del precio Take Profit, pasada por referencia.

expiration

[in][out] Variable de la hora de vencimiento, pasada por referencia.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OpenShortParams

Establece los parámetros de apertura de la posición corta.

```
virtual bool  OpenShortParams(
   double&    price,          // precio
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // vencimiento
   )
```

Parámetros

price

[in][out] Variable del precio, pasada por referencia.

sl

[in][out] Variable del precio Stop Loss, pasada por referencia.

tp

[in][out] Variable del precio Take Profit, pasada por referencia.

expiration

[in][out] Variable de la hora de vencimiento, pasada por referencia.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CheckCloseLong

Comprueba las condiciones de cierre de la posición larga.

```
virtual bool  CheckCloseLong(
   double&  price      // precio
   )
```

Parámetros

price

[in][out] Variable para cerrar el precio, pasada por referencia.

Valor devuelto

true si la condición se satisface, false en otro caso.

© 2000-2025, MetaQuotes Ltd.

CheckCloseShort

Comprueba las condiciones de cierre de la posición corta.

```
virtual bool  CheckCloseShort(
   double&  price      // precio
   )
```

Parámetros

price

[in][out] Variable para cerrar el precio, pasada por referencia.

Valor devuelto

true si la condición se satisface, false en otro caso.

© 2000-2025, MetaQuotes Ltd.

CloseLongParams

Establece los parámetros de cierre de la posición larga.

```
virtual bool  CloseLongParams(
   double&    price          // precio
   )
```

Parámetros

price

[in][out] Variable para cerrar el precio, pasada por referencia.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CloseShortParams

Establece los parámetros de cierre de la posición corta.

```
virtual bool  CloseShortParams(
   double&    price          // precio
   )
```

Parámetros

price

[in][out] Variable para cerrar el precio, pasada por referencia.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CheckReverseLong

Comprueba las condiciones de inversión de la posición larga.

```
virtual bool  CheckReverseLong(
   double&    price,          // precio
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // vencimiento
   )
```

Parámetros

price

[in][out] Variable del precio, pasada por referencia.

sl

[in][out] Variable del precio Stop Loss, pasada por referencia.

tp

[in][out] Variable del precio Take Profit, pasada por referencia.

expiration

[in][out] Variable de la hora de vencimiento, pasada por referencia.

Valor devuelto

true si la condición se satisface, false en otro caso.

© 2000-2025, MetaQuotes Ltd.

CheckReverseShort

Comprueba las condiciones de inversión de la posición corta.

```
virtual bool  CheckReverseShort(
   double&    price,          // precio
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // vencimiento
   )
```

Parámetros

price

[in][out] Variable para la inversión del precio, pasada por referencia.

sl

[in][out] Variable del precio Stop Loss, pasada por referencia.

tp

[in][out] Variable del precio Take Profit, pasada por referencia.

expiration

[in][out] Variable de la hora de vencimiento, pasada por referencia.

Valor devuelto

true si la condición se satisface, false en otro caso.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderLong

Comprueba las condiciones para modificar los parámetros de la orden pendiente de compra.

```
virtual bool  CheckTrailingOrderLong(
   COrderInfo*     order,          // orden
   double&         price           // precio
   )
```

Parámetros

order

[in] Puntero al objeto de la clase COrderInfo.

price

[in][out] Variable de tipo precio Stop Loss.

Valor devuelto

true si la condición se satisface, false en otro caso.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderShort

Comprueba las condiciones para modificar los parámetros de la orden pendiente de venta.

```
virtual bool  CheckTrailingOrderShort(
   COrderInfo*     order,          // orden
   double&         price           // precio
   )
```

Parámetros

order

[in] Puntero al objeto de la clase COrderInfo.

price

[in][out] Variable de tipo precio Stop Loss.

Valor devuelto

true si la condición se satisface, false en otro caso.

© 2000-2025, MetaQuotes Ltd.

LongCondition

Comprueba las condiciones de apertura de la posición larga.

```
virtual int  LongCondition()
```

Valor devuelto

Si las condiciones se satisfacen, devuelve un valor de 1 a 100 (dependiendo de la "fuerza" de la señal), si no hay ninguna señal para abrir una posición larga, devuelve 0.

Nota

El método LongCondition() de la clase base no comprueba las condiciones de apertura de posiciones largas y siempre devuelve 0.

© 2000-2025, MetaQuotes Ltd.

ShortCondition

Comprueba las condiciones de apertura de la posición corta.

```
virtual int  ShortCondition()
```

Valor devuelto

Si las condiciones se satisfacen, devuelve un valor entre 1 y 100 (dependiendo de la "fuerza" de la señal), si no hay ninguna señal para abrir una posición corta, devuelve 0.

Nota

El método ShortCondition() de la clase base no comprueba las condiciones de apertura de la posición corta y siempre devuelve 0.

© 2000-2025, MetaQuotes Ltd.

Direction

Devuelve el valor de la dirección "ponderada".

```
virtual double  Direction()
```

Valor devuelto

Devuelve un valor > 0 cuando la dirección es (probablemente) hacia arriba y un valor < 0 cuando es hacia abajo. El valor absoluto depende de la "fuerza" de la señal.

Nota

Si se utilizan los filtros, el resultado dependerá de los mismos.

© 2000-2025, MetaQuotes Ltd.

CExpertTrailing

CExpertTrailing es la clase base de los algoritmos de trailing, no hace nada sino que proporciona las interfaces.

Cómo utilizarla:

1. Prepare un algoritmo de trailing;
2. Cree su propia clase de trailing, haciéndola heredar de la clase CExpertTrailing;
3. Sobreescriba los métodos virtuales de su clase, escribiendo sus propios algoritmos.

Hay ejemplos disponibles de clases de trailing en la carpeta Expert\Trailing\.

Descripción

CExpertTrailing es la clase base para implementar algoritmos de trailing stop.

Declaración

```
class CExpertTrailing : public CExpertBase
```

Título

```
#include <Expert\ExpertTrailing.mqh>
```

Jerarquía de herencia

CObject

CExpertBase

CExpertTrailing

Descendientes directos

CTrailingFixedPips, CTrailingMA, CTrailingNone, CTrailingPSAR

Métodos de la clase

Comprobación de las condiciones de Trailing Stop	
virtual CheckTrailingStopLong	Comprueba las condiciones para modificar los parámetros de la posición larga
virtual CheckTrailingStopShort	Comprueba las condiciones para modificar los parámetros de la posición corta

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Comprueba las condiciones para modificar los parámetros de la posición larga.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // puntero
   double&         sl,           // Stop Loss
   double&         tp            // Take Profit
   )
```

Parámetros

position

[in] Puntero al objeto de la clase CPositionInfo.

sl

[in][out] Variable del precio Stop Loss, pasada por referencia.

tp

[in][out] Variable del precio Take Profit, pasada por referencia.

Valor devuelto

true si la condición se satisface, false en otro caso.

Nota

El método CheckTrailingStopLong(...) de la clase base siempre devuelve false.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Comprueba las condiciones para modificar los parámetros de la posición corta.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // puntero
   double&         sl,           // Stop Loss
   double&         tp            // Take Profit
   )
```

Parámetros

position

[in] Puntero al objeto de la clase CPositionInfo.

sl

[in][out] Variable del precio Stop Loss, pasada por referencia.

tp

[in][out] Variable del precio Take Profit, pasada por referencia.

Valor devuelto

true si la condición se satisface, false en otro caso.

Nota

El método CheckTrailingStopShort(...) de la clase base siempre devuelve false.

© 2000-2025, MetaQuotes Ltd.

CExpertMoney

CExpertMoney es la clase base de los algoritmos de gestión del dinero y gestión del riesgo.

Descripción

CExpertMoney es una clase base para la implementación de clases de gestión del dinero y gestión del riesgo.

Declaración

```
class CExpertMoney : public CObject
```

Título

```
#include <Expert\ExpertMoney.mqh>
```

Jerarquía de herencia

CObject

CExpertBase

CExpertMoney

Descendientes directos

CMoneyFixedLot, CMoneyFixedMargin, CMoneyFixedRisk, CMoneyNone, CMoneySizeOptimized

Métodos de la clase

Acceso a datos protegidos	
Percent	Establece el parámetro "Porcentaje de riesgo"
Inicialización	
virtual ValidationSettings	Comprueba las configuraciones
Comprobación de las condiciones de trading	
virtual CheckOpenLong	Obtiene el volumen de la posición larga
virtual CheckOpenShort	Obtiene el volumen de la posición corta
virtual CheckReverse	Obtiene el volumen de la posición inversa
virtual CheckClose	Comprueba las condiciones para cerrar la operación abierta

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2025, MetaQuotes Ltd.

Percent

Establece el valor del parámetro "Porcentaje de riesgo".

```
void  Percent(
   double  percent      // porcentaje de riesgo
   )
```

Parámetros

percent

[in] Porcentaje de riesgo.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Comprueba las configuraciones.

```
virtual bool  ValidationSettings()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

El método ValidationSettings() de la clase base siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Obtiene el volumen de la posición larga.

```
virtual double  CheckOpenLong(
   double  price,     // precio
   double  sl         // Stop Loss
   )
```

Parámetros

price

[in] Precio de apertura de la posición larga.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición larga.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Obtiene el volumen de la posición corta.

```
virtual double  CheckOpenShort(
   double  price,     // precio
   double  sl         // Stop Loss
   )
```

Parámetros

price

[in] Precio de apertura de la posición corta.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición corta.

© 2000-2025, MetaQuotes Ltd.

CheckReverse

Obtiene el volumen de la posición inversa.

```
virtual double  CheckReverse(
   CPositionInfo*  position,     // puntero
   double          sl            // Stop Loss
   )
```

Parámetros

position

[in] Puntero al objeto de la clase CPositionInfo.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición inversa.

© 2000-2025, MetaQuotes Ltd.

CheckClose

Comprueba las condiciones para cerrar la posición abierta.

```
virtual double  CheckClose()
```

Valor devuelto

true si la condición se satisface, false en otro caso.

© 2000-2025, MetaQuotes Ltd.

Módulos de señales de trading

El terminal cliente incluye en el "Asistente MQL5" un conjunto de módulos de señales de trading prefabricados. Al crear un Asesor Experto con el Asistente MQL5, se puede utilizar cualquier combinación de módulos de señales de trading (hasta 64). La decisión final sobre una operación se realiza en base al análisis complejo de las señales obtenidas en los módulos incluidos. La descripción detallada del mecanismo de toma de decisiones comerciales se describe a continuación.

El terminal cliente incluye los siguientes módulos de señales:

- Señales del indicador Accelerator Oscillator
- Señales del indicador Media Móvil Adaptativa
- Señales del indicador Awesome Oscillator
- Señales del oscilador Bears Power
- Señales del oscilador Bulls Power
- Señales del oscilador Commodity Channel Index
- Señales del oscilador DeMarker
- Señales del indicador Media Móvil Exponencial Doble
- Señales del indicador Envelopes
- Señales del indicador Media Móvil Adaptativa Fractal
- Señales del Filtro de Tiempo Intradía
- Señales del oscilador MACD
- Señales del indicador Media Móvil
- Señales del indicador SAR Parabólico
- Señales del oscilador Índice de Fuerza Relativa
- Señales del oscilador Índice de Vigor Relativo
- Señales del oscilador Estocástico
- Señales del oscilador Media Exponencial Triple
- Señales del indicador Media Móvil Exponencial Triple
- Señales del oscilador Rango Porcentual de Williams

Mecanismo de toma de decisiones de trading en base a los módulos de señales

El mecanismo de toma de decisiones comerciales se puede resumir en la siguiente lista de principios básicos:

- Cada módulo de señales tiene su propio conjunto de módulos del mercado, esto es, una cierta combinación de precios y valores de un indicador.
- Cada modelo de mercado tiene una importancia que puede variar en el rango de 1 a 100. Cuanto mayor es la importancia, más fuerte es el modelo.
- Cada modelo genera una previsión de la dirección del movimiento del precio.
- La previsión de un módulo es el resultado de buscar modelos integrados, y se genera como un número comprendido entre -100 y 100. El signo determina la dirección del movimiento previsto (un

© 2000-2025, MetaQuotes Ltd.

signo negativo significa que el precio caerá, y un signo positivo significa que el precio subirá). El valor absoluto se corresponde con la fuerza del mejor modelo encontrado.

- El pronóstico del módulo se envía a la "votación" final con un coeficiente de peso entre 0 y 1, especificado en su configuración ("Peso").
- El resultado de la votación es un número entre -100 y 100, donde el signo determina la dirección del movimiento previsto, y el valor absoluto caracteriza la fuerza de la señal. Se calcula como la media aritmética de las previsiones ponderadas de todos los módulos de señales.

Cada Asesor Experto generado tiene dos configuraciones ajustables: los niveles de umbral de apertura y cierre de una posición (ThresholdOpen y ThresholdClose), que pueden estar comprendidos entre 0 y 100. Si la fuerza de la señal final supera un nivel umbral, se realiza una operación de trading que se corresponde con el signo de la señal.

Ejemplos

Considere un Asesor Experto con los siguientes niveles de umbral: ThresholdOpen=20 y ThresholdClose=90. En la toma de decisiones de las operaciones participan dos módulos de señales: el módulo MA, que pesa 0.4, y el módulo Estocástico, que pesa 0.8. Analicemos dos variantes de las señales de trading obtenidas:

Variante 1.

El precio cruza la MA creciente hacia arriba. Este caso corresponde a uno de los modelos de mercado implementados en el módulo MA. Este modelo implica una subida de precio. Su importancia es igual a 100. Al mismo tiempo, el oscilador Estocástico da la vuelta hacia abajo y forma una divergencia con el precio. Este caso corresponde a uno de los modelos implementados en el módulo Estocástico. Este modelo implica una caída del precio. El peso de este modelo es 80.

Calculemos el resultado de la "votación" final. La tasa obtenida en el módulo MA se calcula como 0.4 * 100 = 40. El valor del módulo estocástico se calcula como 0.8 * (-80) = -64. El valor final se calcula como la media aritmética de estas dos tasas: (40 - 64)/2 = -12. El resultado de la votación es la señal de venta con fuerza relativa igual a 12. El nivel de umbral, igual a 20, no se alcanza. Por lo tanto no se lleva a cabo ninguna operación de trading.

Variante 2.

El precio cruza la MA creciente hacia abajo. Este caso corresponde a uno de los modelos implementados en el módulo MA. Este modelo implica una subida de precio. Su importancia es igual a 10. Al mismo tiempo, el oscilador Estocástico da la vuelta hacia abajo y forma una divergencia con el precio. Este caso corresponde a uno de los modelos implementados en el módulo Estocástico. Este modelo implica una caída del precio. El peso de este modelo es 80.

Calculemos el resultado de la "votación" final. La tasa obtenida con el módulo MA se calcula como 0.4 * 10 = 4. El valor del módulo estocástico se calcula como 0.8 * (-80) = -64. El valor final se calcula como la media aritmética de estas dos tasas: (4 - 64)/2 = -30. El resultado de la votación es la señal de venta con fuerza relativa igual a 30. Se alcanza el nivel de umbral igual a 20. Por lo tanto el resultado es una señal de apertura de una posición corta.

© 2000-2025, MetaQuotes Ltd.



a) Divergencia del precio y el oscilador Estocástico (variantes 1 y 2).

b) El precio cruza hacia arriba el indicador MA (variante 1).

c) El precio cruza hacia abajo el indicador MA (variante 2).

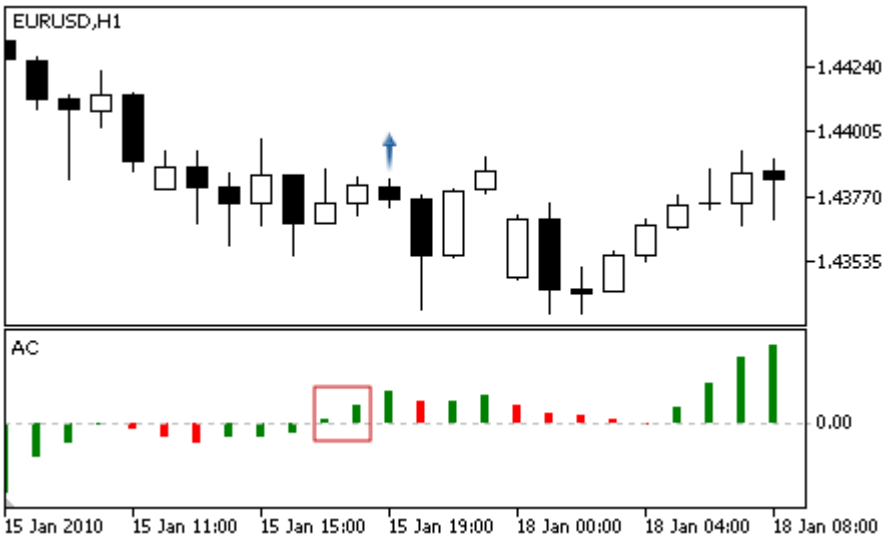
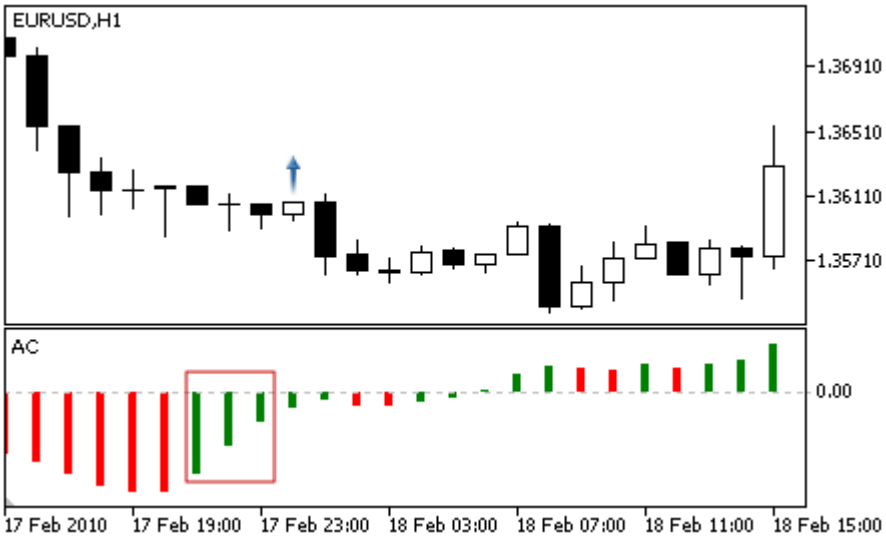
© 2000-2025, MetaQuotes Ltd.

Señales del indicador Accelerator Oscillator

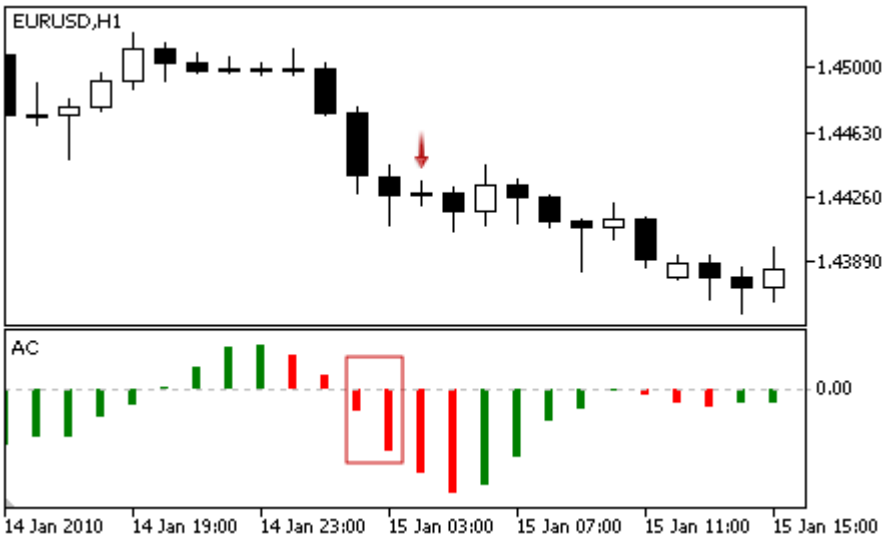
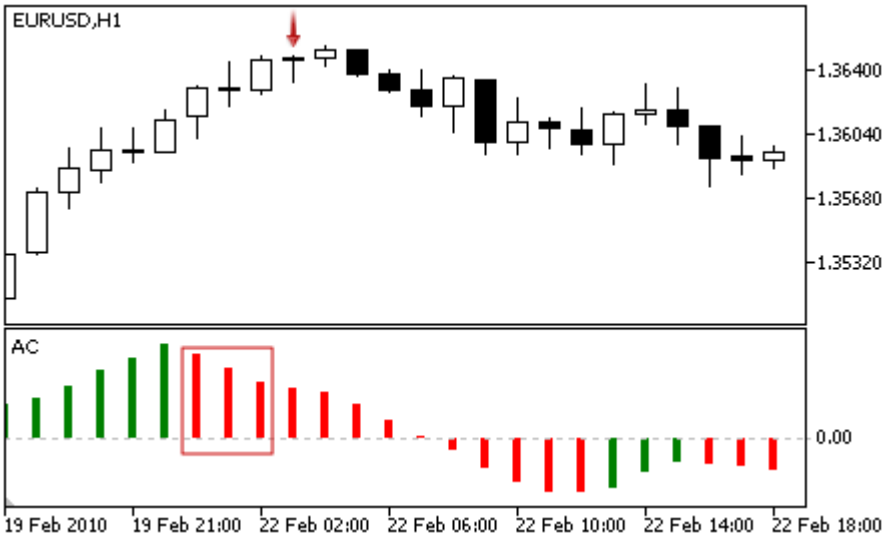
Este módulo se basa en los modelos de mercado del indicador Accelerator Oscillator. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• El valor del indicador es superior a 0 y se incrementa en las barras anteriores analizadas.  • El valor del indicador es inferior a 0 y se incrementa en las barras analizadas anteriores. 
De venta	• El valor del indicador es inferior a 0 y disminuye en las barras analizadas anteriores.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • El valor del indicador es inferior a 0 y disminuye en las barras analizadas anteriores. 
No hay objeciones para comprar	El valor del indicador aumenta en la barra analizada.
No hay objeciones para vender	El valor del indicador disminuye en la barra analizada.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.

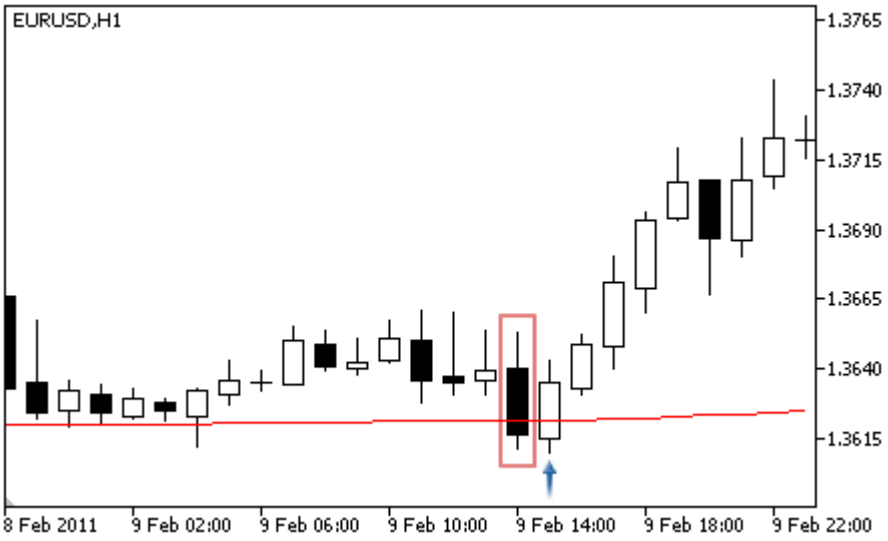
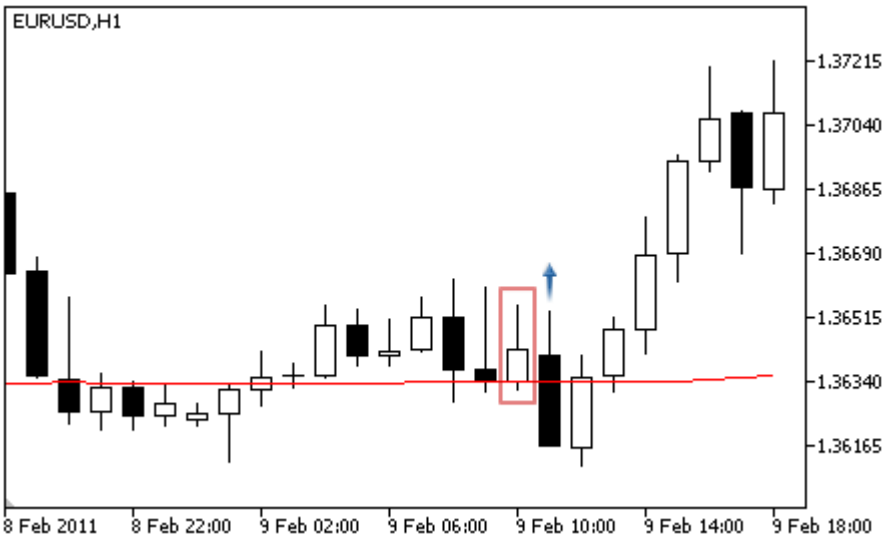
© 2000-2025, MetaQuotes Ltd.

Señales del indicador Media Móvil Adaptativa

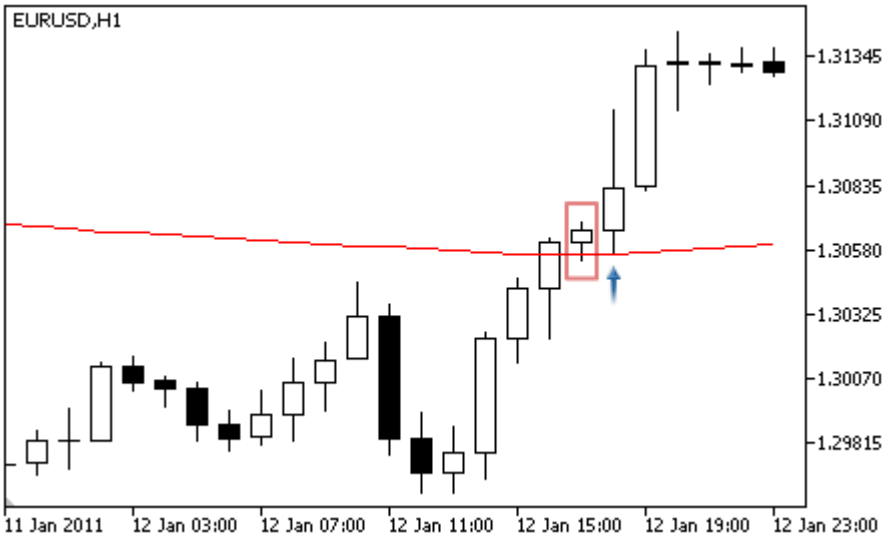
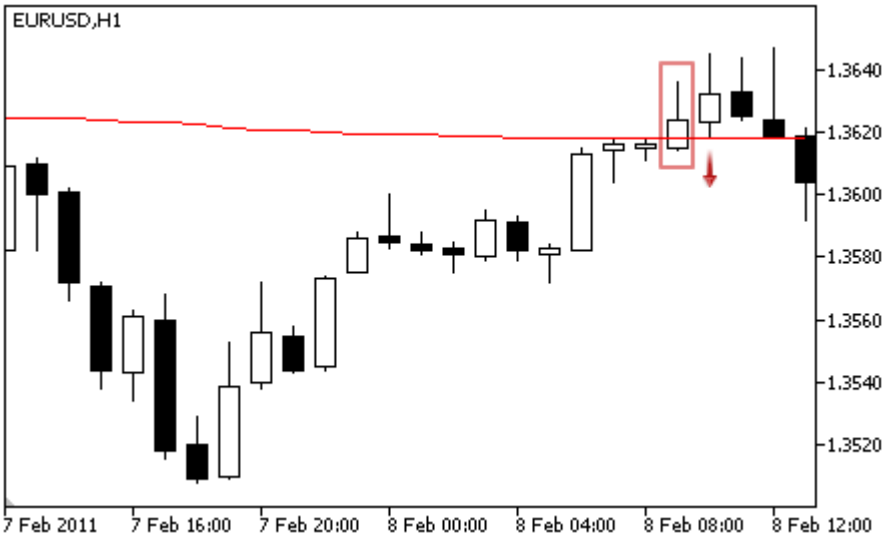
Este módulo se basa en los modelos de mercado del indicador Media Móvil Adaptativa. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

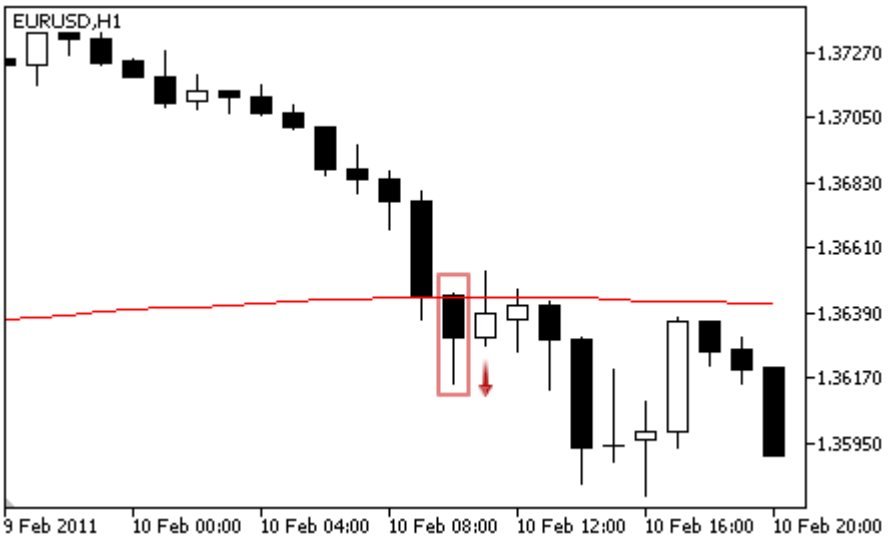
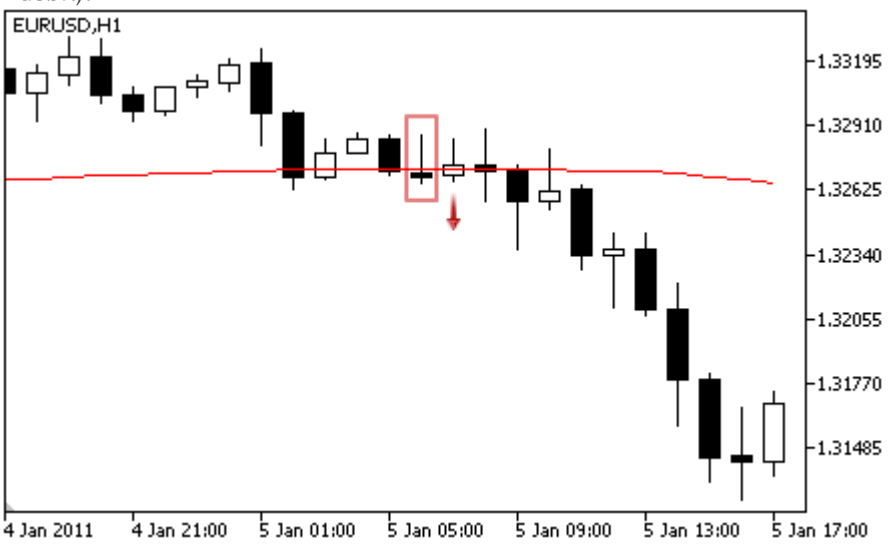
A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• El precio ha cruzado el indicador hacia abajo (el precio de apertura de la barra analizada está por encima del indicador, y el precio de cierre se encuentra por debajo del mismo) y el indicador sube (señal débil).  • **Cruce de la media móvil.** El precio ha cruzado el indicador hacia arriba (el precio de apertura de la barra analizada está por debajo del indicador, y el precio de cierre se encuentra por encima del mismo) y el indicador sube (señal fuerte). 

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	• La sombra inferior de la barra ha cruzado el indicador (los precios de apertura y cierre de la barra analizada están por encima del indicador, y el precio bajo se encuentra por debajo del mismo) y el indicador sube (señal débil). 
De venta	• El precio ha cruzado el indicador hacia arriba (el precio de apertura de la barra analizada está por debajo del indicador, y el precio de cierre se encuentra sobre el mismo) y el indicador cae (señal débil).  • **Cruce de la media móvil.** El precio ha cruzado el indicador hacia abajo (el precio de apertura de la barra analizada está por encima del indicador, y el precio de cierre se encuentra por debajo del mismo) y el indicador cae (señal fuerte).

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • La sombra superior de la barra ha cruzado el indicador (los precios de apertura y cierre de la barra analizada están por debajo del indicador, y el precio alto se encuentra sobre el mismo) y el indicador cae (señal débil). 
No hay objeciones para comprar	El precio está por encima del indicador.
No hay objeciones para vender	El precio está por debajo del indicador.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodMA	Período de promediación del indicador.
Shift	Desplazamiento del indicador a lo largo del eje temporal (en barras).
Method	Método de promediación.
Applied	La serie de precios utilizada para calcular el indicador.

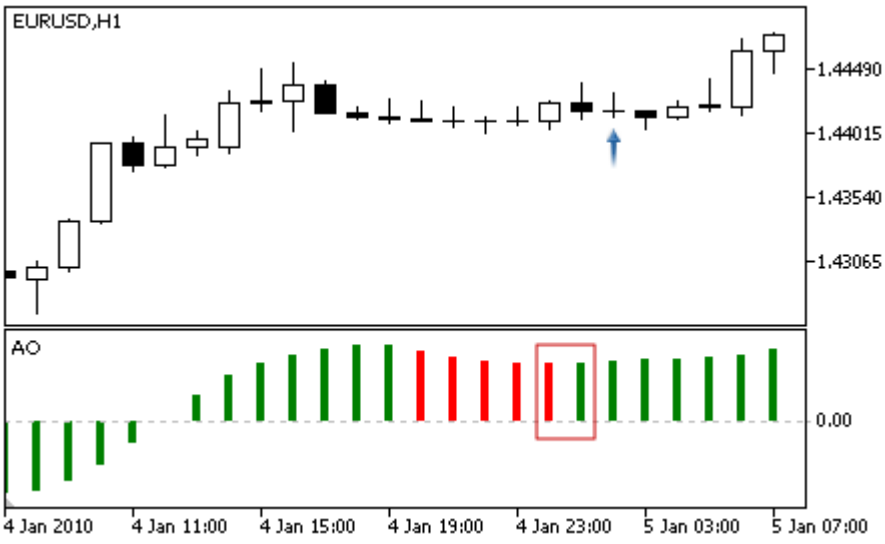
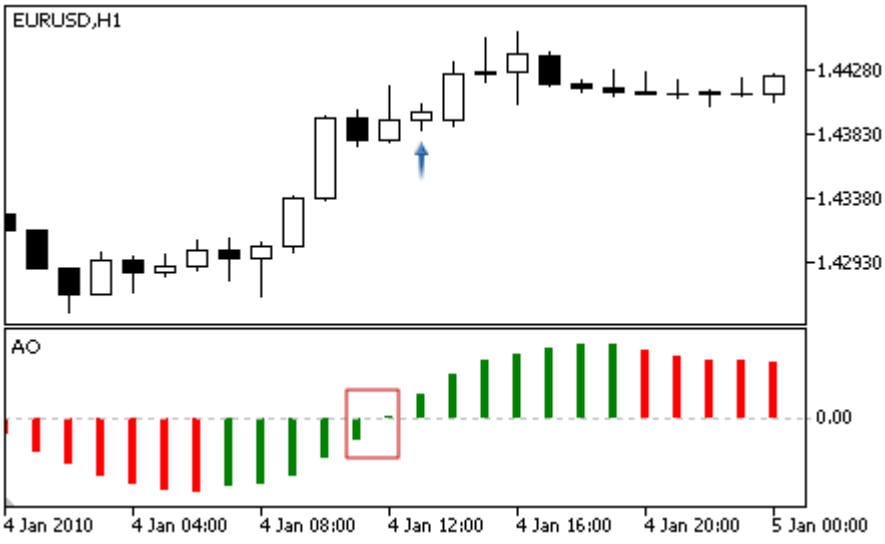
© 2000-2025, MetaQuotes Ltd.

Señales del indicador Awesome Oscillator

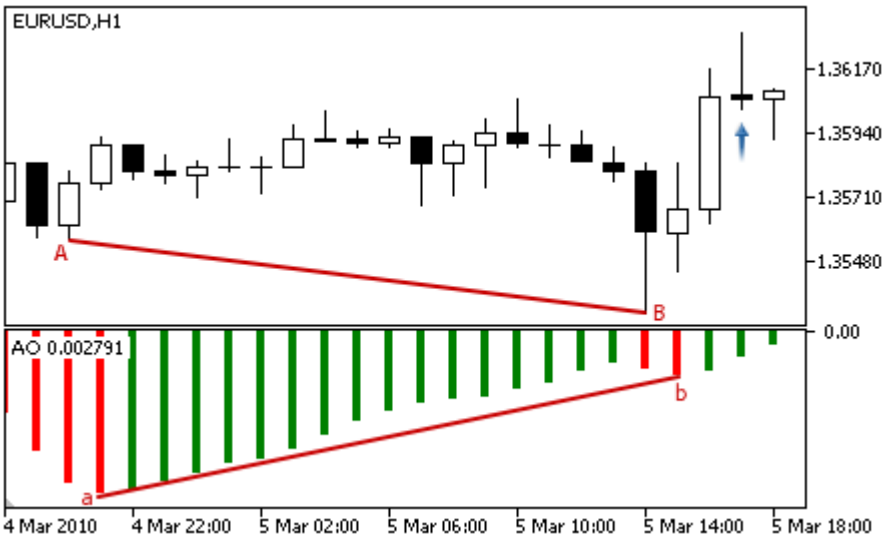
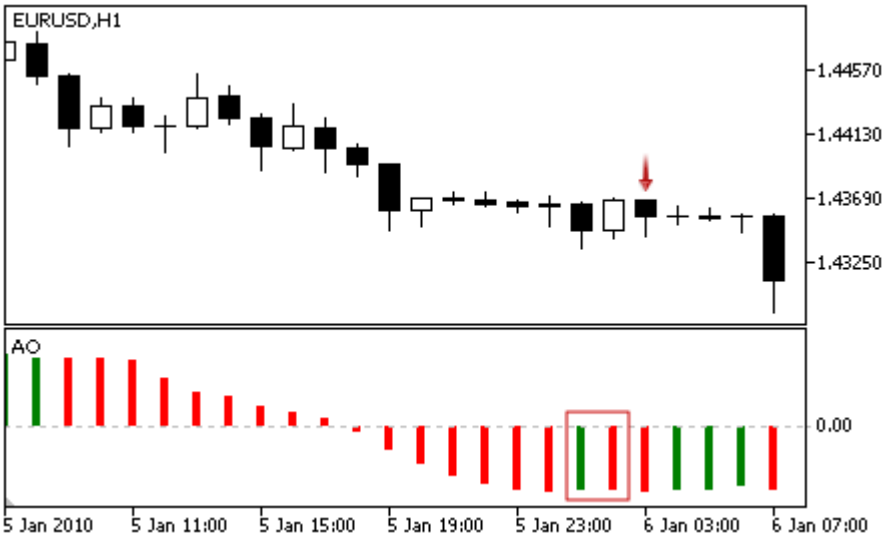
Este módulo de señales se basa en los modelos de mercado del indicador Awesome Oscillator. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

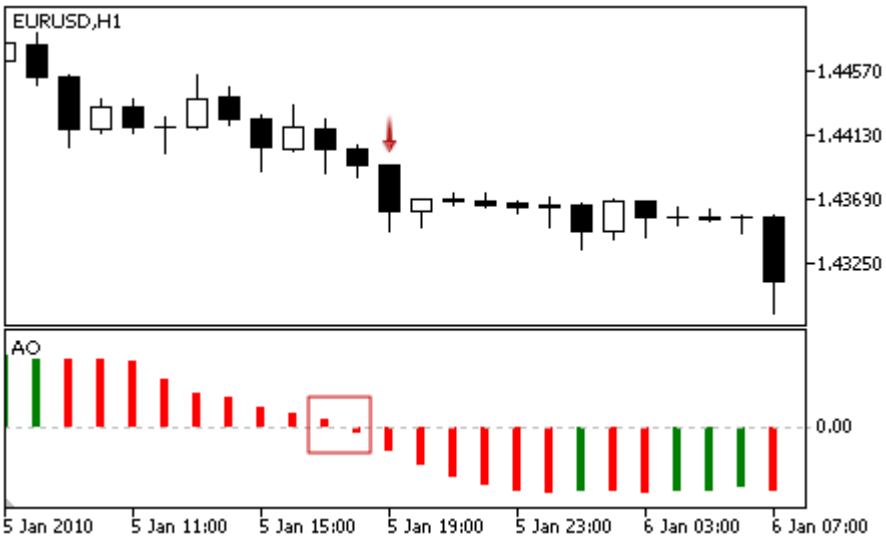
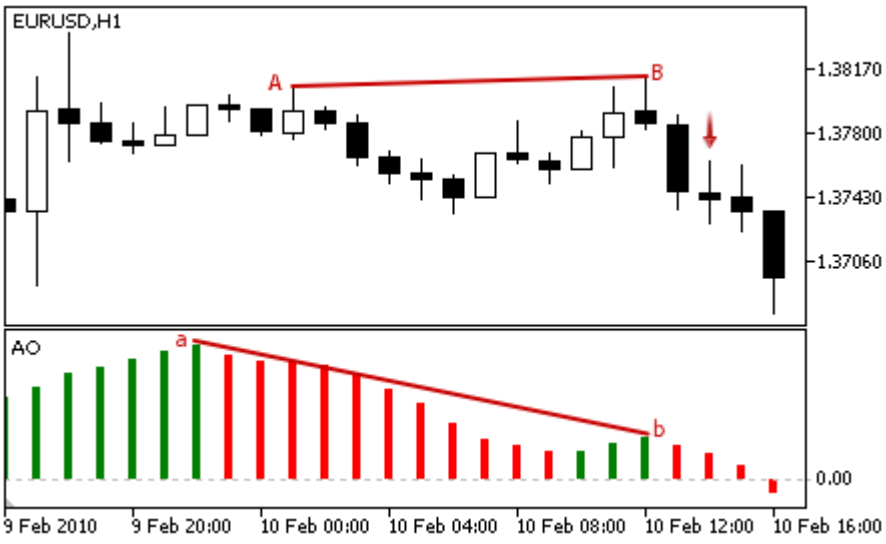
A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• **Platillo** — el valor del indicador en la barra analizada sube, y cae en las barras anteriores; siendo ambos valores mayores que 0.  • **Cruce de la línea cero** — el valor del indicador es superior a 0 en la barra analizada, e inferior a 0 en la barra anterior.  • **Divergencia** — el primer pico inferior analizado por el indicador es menor que el anterior, y el precio del valle correspondiente es más

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	profundo que el anterior. Además, el indicador no debe superar el nivel cero. 
De venta	• **Platillo** — el valor del indicador en la barra analizada cae, y se levanta en las barras anteriores; siendo ambos valores inferiores a 0.  • **Cruce de la línea cero** — el valor del indicador es inferior 0 en la barra analizada, y es superior a 0 en la anterior.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • **Divergencia** — el primer pico analizado del indicador es más bajo que el anterior, y el pico del precio correspondiente es más alto que el anterior. Además, el indicador no debe caer por debajo del nivel cero. 
No hay objeciones para comprar	El valor del indicador aumenta en la barra analizada.
No hay objeciones para vender	El valor del indicador disminuye en la barra analizada.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.

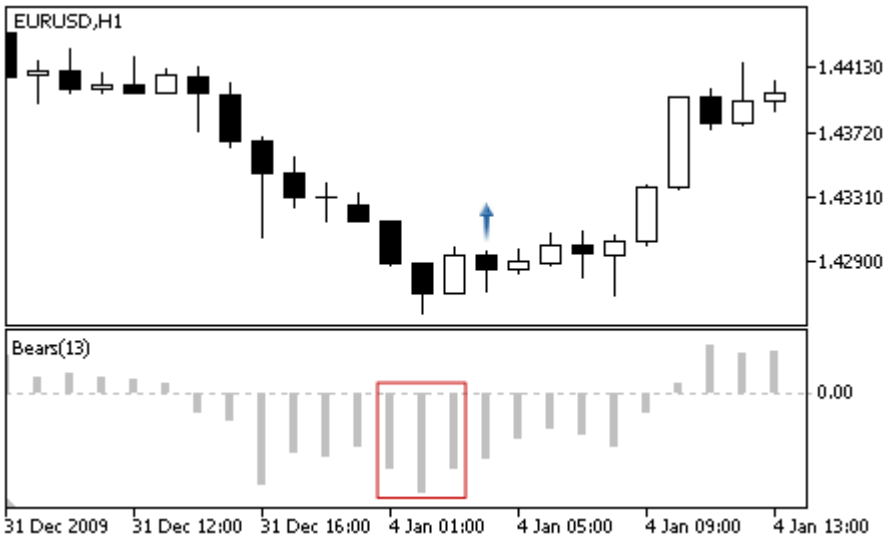
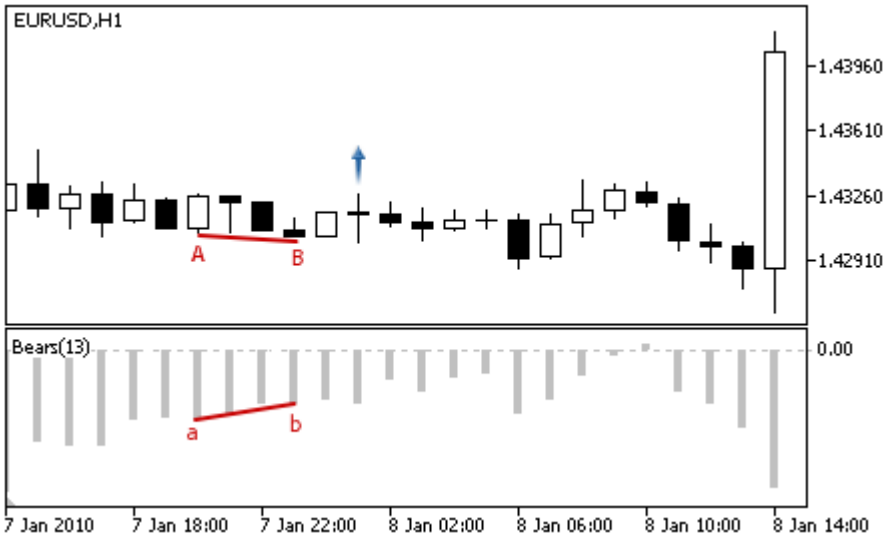
© 2000-2025, MetaQuotes Ltd.

Señales del oscilador Bears Power

Este módulo de señales se basa en los modelos de mercado del oscilador Bears Power. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• **Inversión** — el oscilador se vuelve hacia arriba y su valor en la barra analizada es inferior a 0.  • **Divergencia** — el primer pico inferior analizado por el oscilador es mayor que el anterior, y el pico inferior del precio correspondiente es más bajo que el anterior. Además, el oscilador no debe elevarse por encima del nivel cero. 
De venta	Sin señales de venta.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
No hay objeciones para comprar	El valor del oscilador es menor que 0.
No hay objeciones para vender	Sin señales.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodBears	Período de cálculo del oscilador.

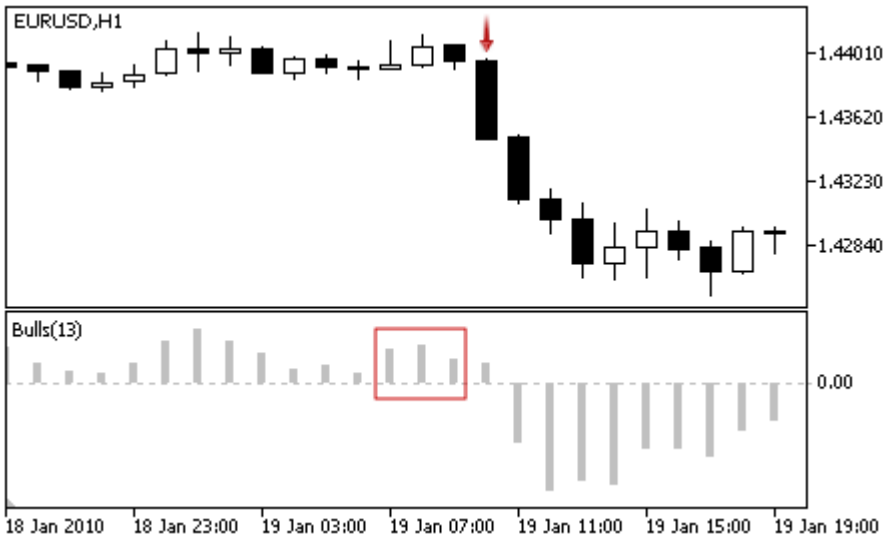
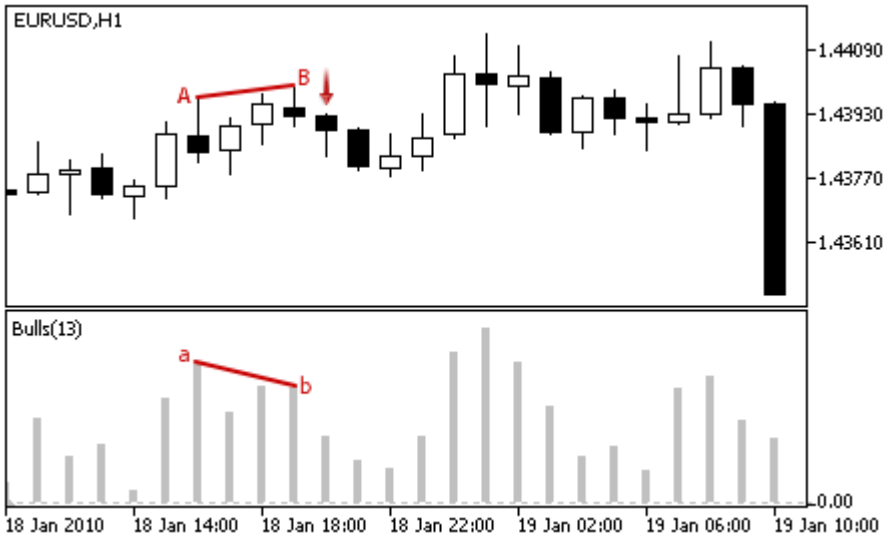
© 2000-2025, MetaQuotes Ltd.

Señales del oscilador Bulls Power

Este módulo de señales se basa en los modelos de mercado del oscilador Bulls Power. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	Sin señales de compra.
De venta	• **Inversión** — el oscilador se gira hacia abajo y su valor en la barra analizada es superior a 0.  • **Divergencia** — el primer pico analizado del oscilador es menor que el anterior, y el pico del precio correspondiente es más alto que el anterior. Además, el oscilador no debe caer por debajo del nivel cero. 

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
No hay objeciones para comprar	Sin señales.
No hay objeciones para vender	El valor del oscilador es mayor que 0.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodBulls	Período de cálculo del oscilador.

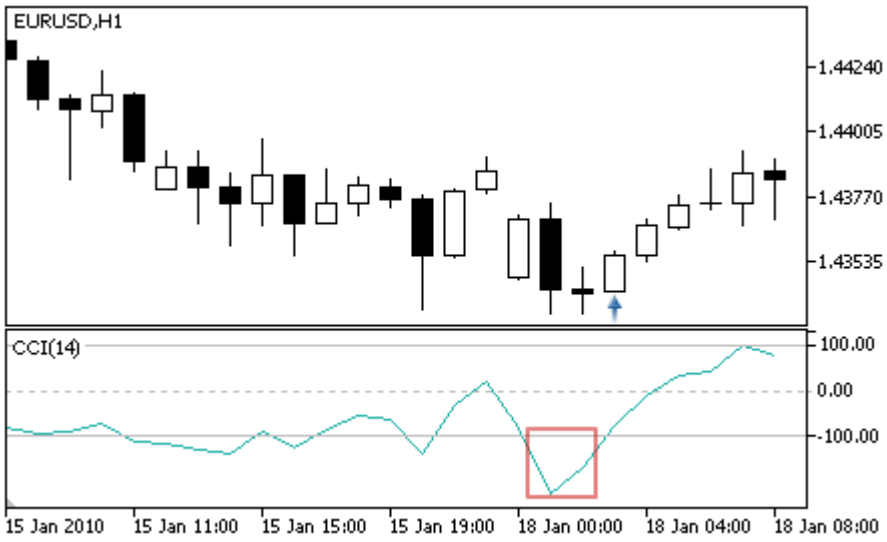
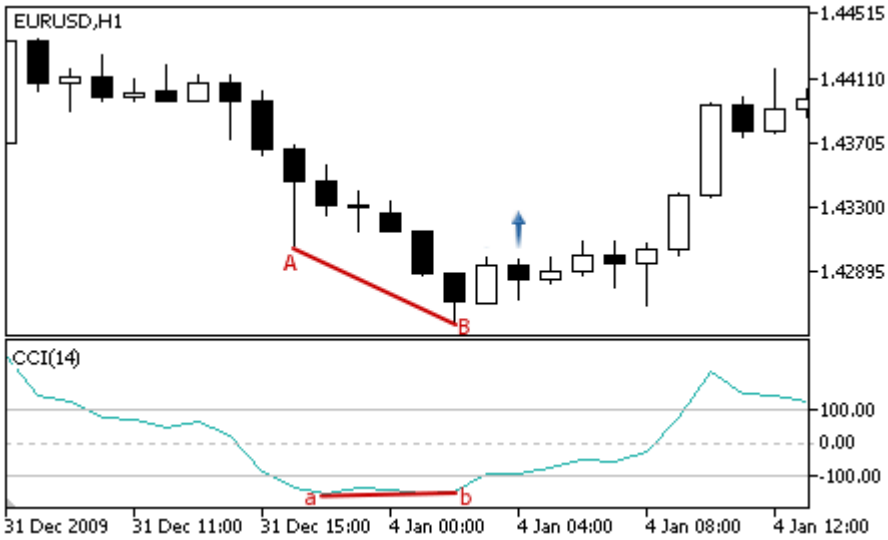
© 2000-2025, MetaQuotes Ltd.

Señales del oscilador Commodity Channel Index

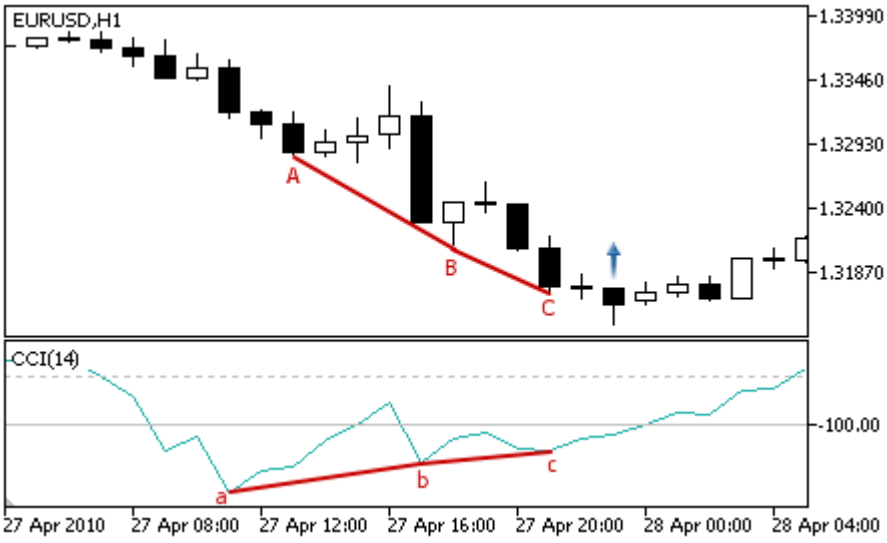
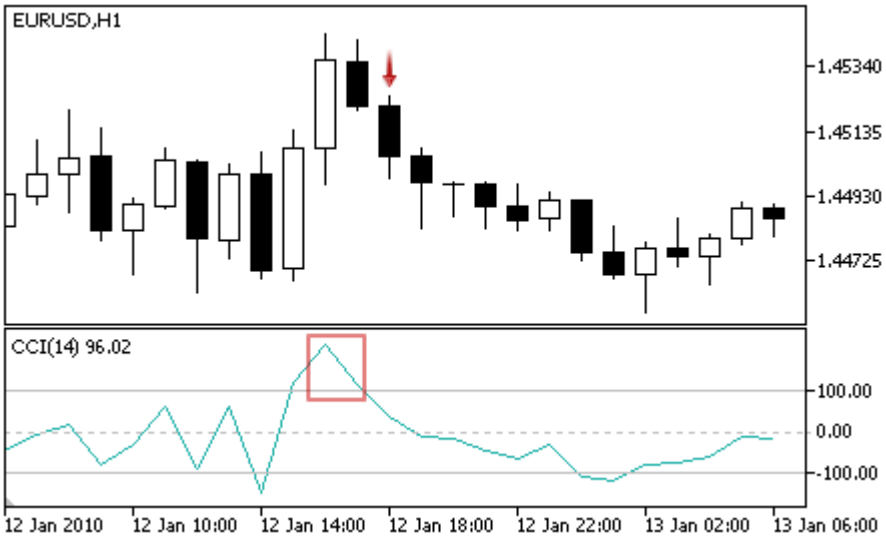
Este módulo de señales se basa en los modelos de mercado del oscilador Commodity Channel Index. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

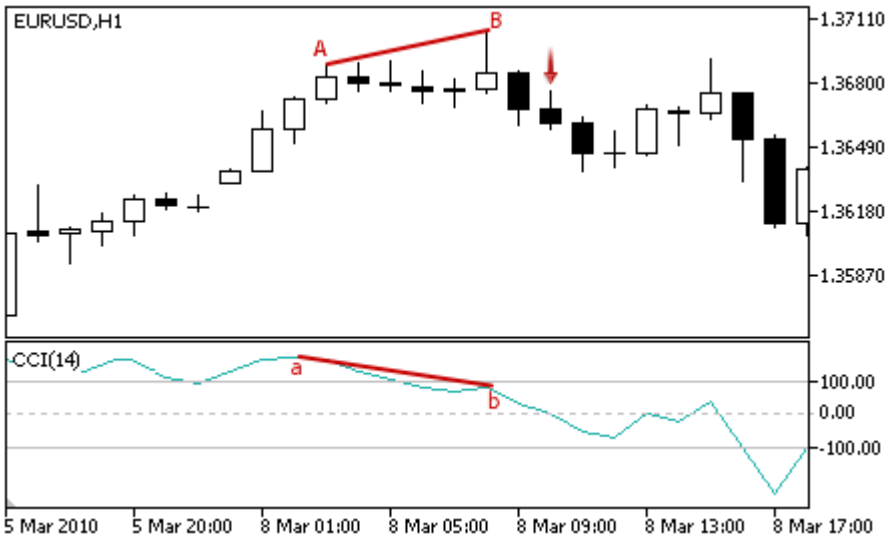
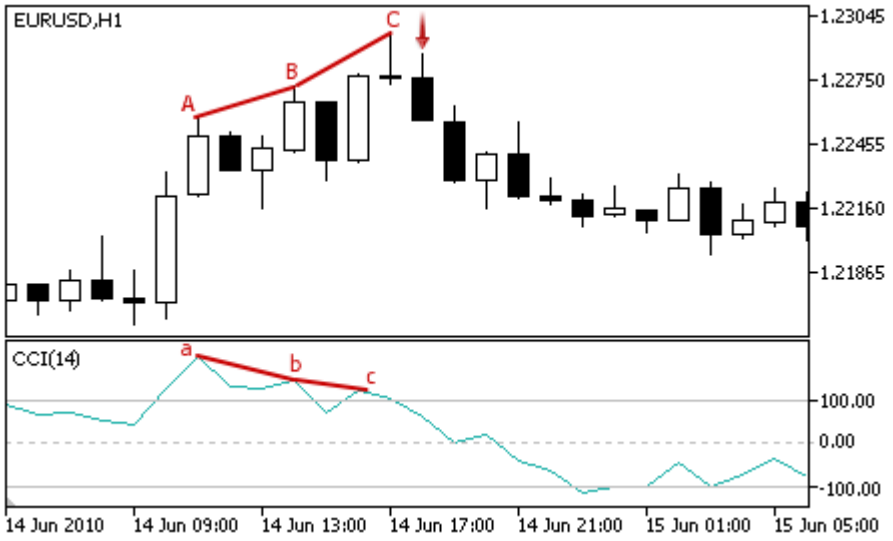
A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• **Inversión por detrás del nivel de sobreventa.** — el oscilador se gira hacia arriba y su valor en la barra analizada está por detrás del nivel de sobreventa (el valor predeterminado es -100).  • **Divergencia** — el primer pico inferior analizado por el oscilador es mayor que el anterior, y el pico inferior del precio correspondiente es más bajo que el anterior. 

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	• **Divergencia doble** — el oscilador forma tres picos inferiores consecutivos, cada uno mayor que el anterior; y el precio forma los tres picos inferiores correspondientes, siendo cada uno de ellos menor que el anterior. 
De venta	• **Inversión por detrás del nivel de sobrecompra** — el oscilador se gira hacia abajo y su valor en la barra analizada está por detrás del nivel de sobrecompra (el valor por defecto es 100).  • **Divergencia** — el primer pico analizado del oscilador es menor que el anterior, y el pico del precio correspondiente es más alto que el pico anterior.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • **Divergencia doble** — el oscilador forma tres picos consecutivos, cada uno menor que el anterior; y el precio forma tres picos correspondientes, siendo cada uno de ellos mayor que el anterior. 
No hay objeciones para comprar	El valor del oscilador crece en la barra analizada.
No hay objeciones para vender	El valor del oscilador cae en la barra analizada.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodCCI	Período de cálculo del oscilador.
Applied	La serie de precios utilizada en el cálculo del oscilador.

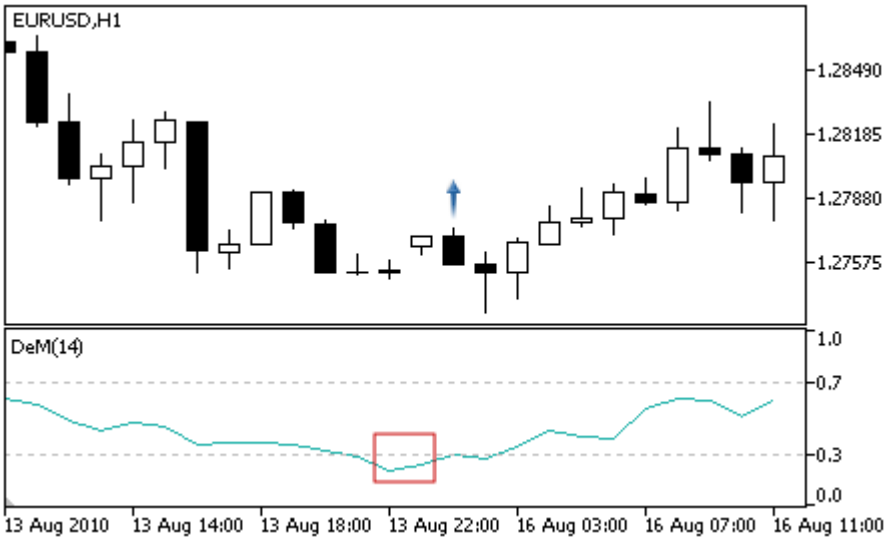
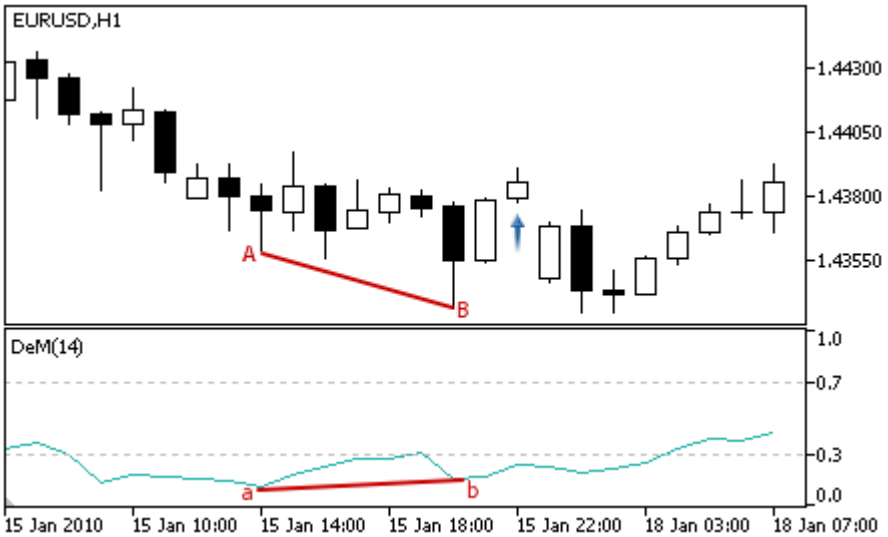
© 2000-2025, MetaQuotes Ltd.

Señales del oscilador DeMarker

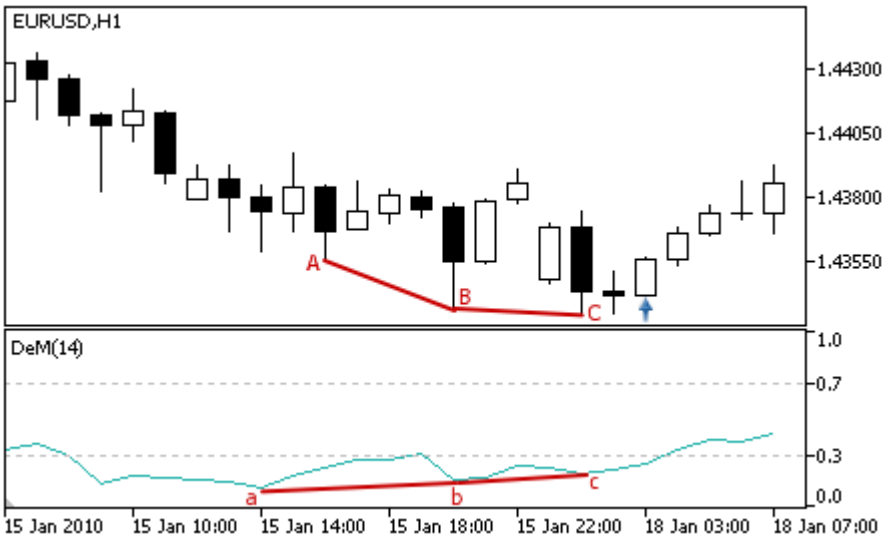
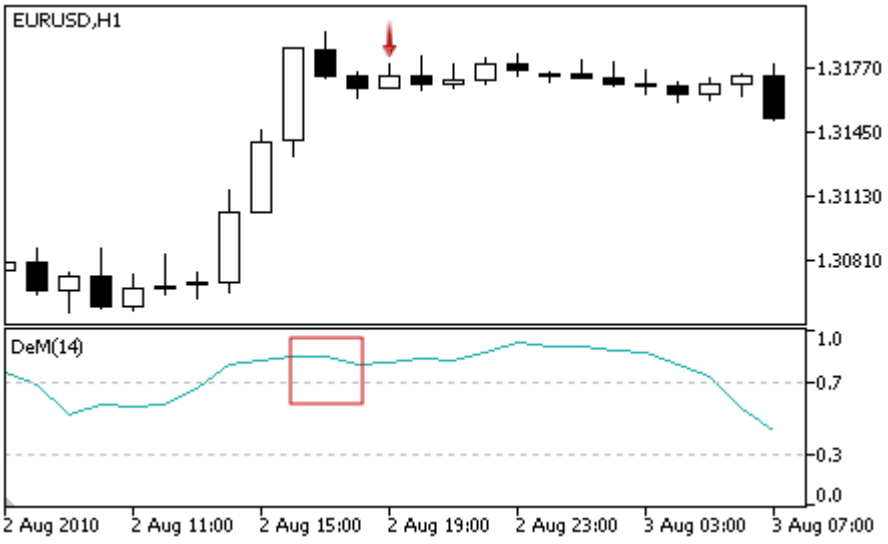
Este módulo de señales se basa en los modelos de mercado del oscilador DeMarker. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

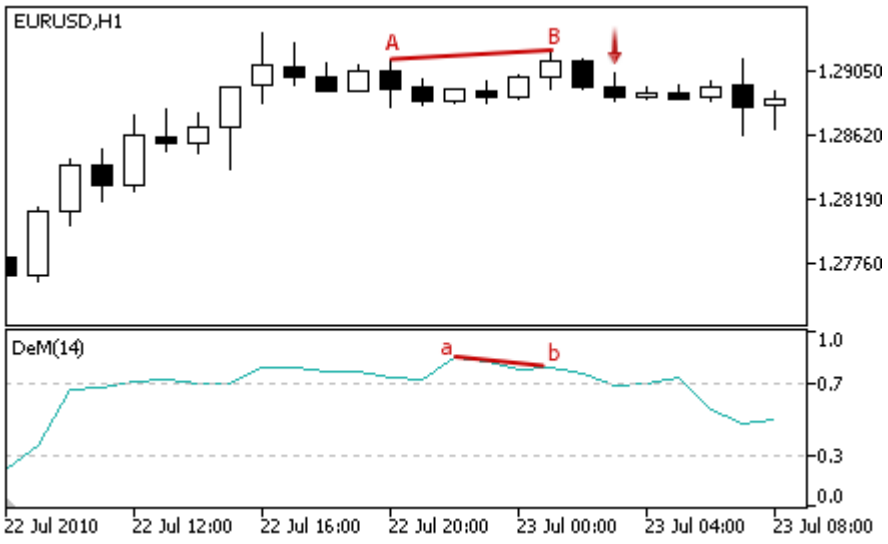
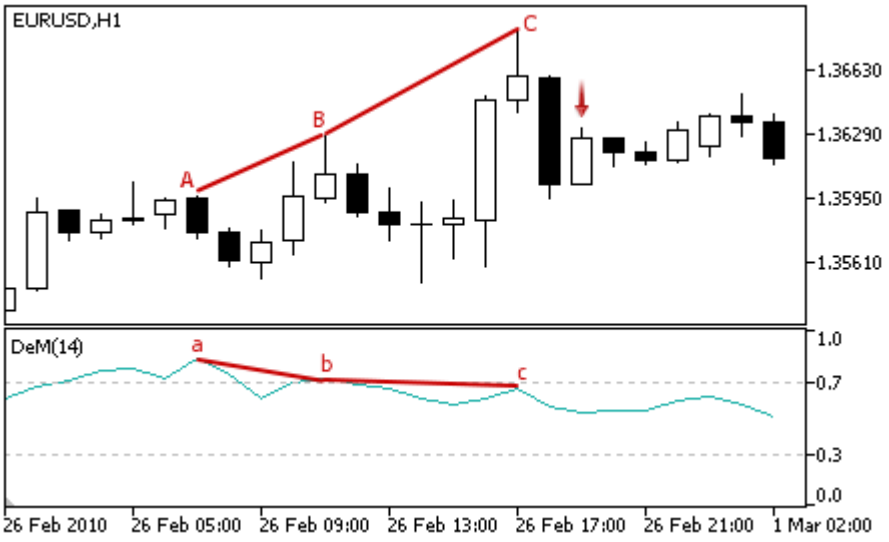
A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• **Inversión por detrás del nivel de sobreventa** — el oscilador se gira hacia arriba, y su valor en la barra analizada está por detrás del nivel de sobreventa (el valor predeterminado es 0.3).  • **Divergencia** — el primer pico inferior analizado por el oscilador es mayor que el anterior, y el pico inferior del precio correspondiente es más bajo que el anterior.  • **Divergencia doble** — el oscilador forma tres picos inferiores consecutivos, cada uno mayor que el anterior; y el precio forma los tres

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	picos inferiores correspondientes, siendo cada uno de ellos menor que el anterior. 
De venta	• **Inversión por detrás del nivel de sobrecompra** — el oscilador se gira hacia abajo y su valor en la barra analizada está por detrás del nivel de sobrecompra (el valor predeterminado es 0.7).  • **Divergencia** — el primer pico analizado del oscilador es menor que el anterior, y el pico del precio correspondiente es más alto que el pico anterior.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • **Divergencia doble** — el oscilador forma tres picos consecutivos, cada uno menor que el anterior; y el precio forma tres picos correspondientes, siendo cada uno de ellos mayor que el anterior. 
No hay objeciones para comprar	El valor del oscilador crece en la barra analizada.
No hay objeciones para vender	El valor del oscilador cae en la barra analizada.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodDeM	Período de cálculo del oscilador.

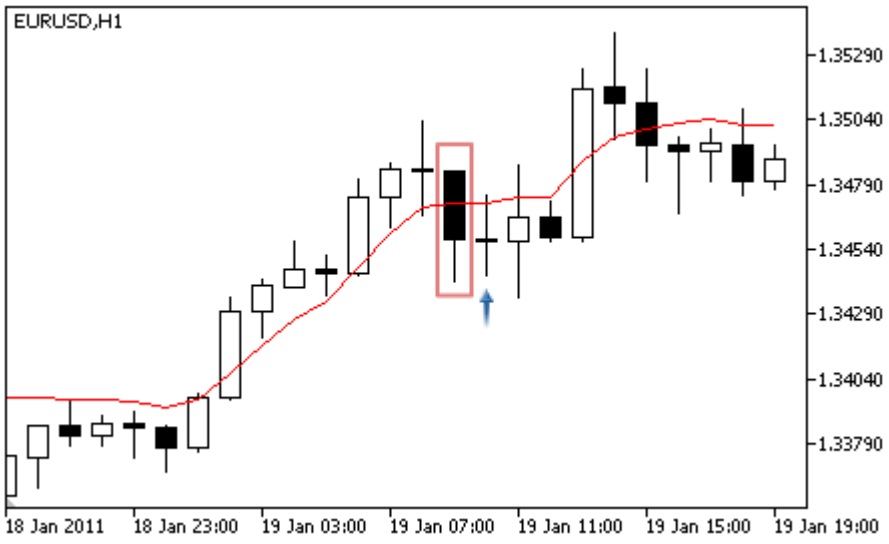
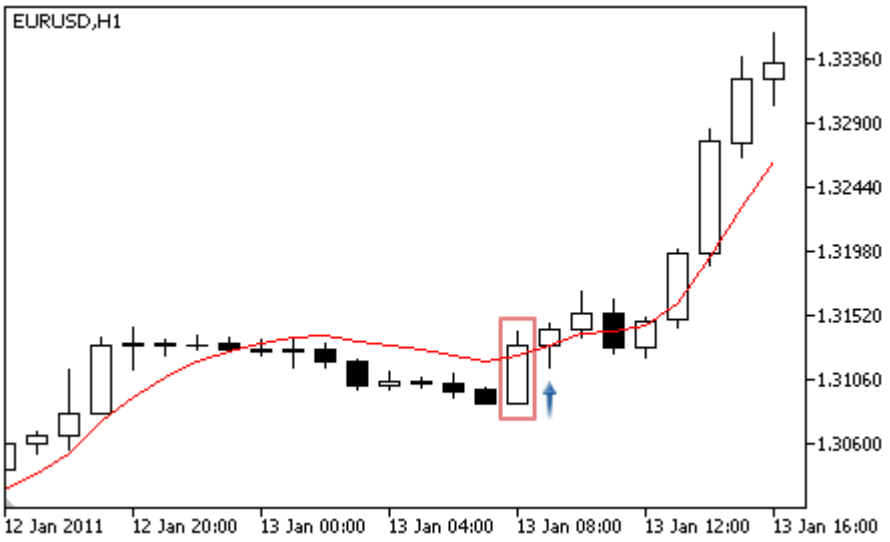
© 2000-2025, MetaQuotes Ltd.

Señales del indicador Media Móvil Exponencial Doble

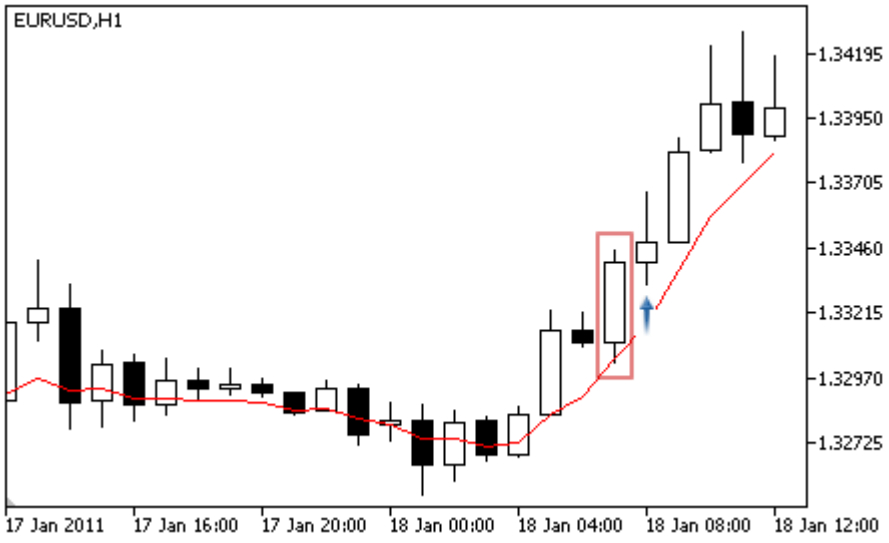
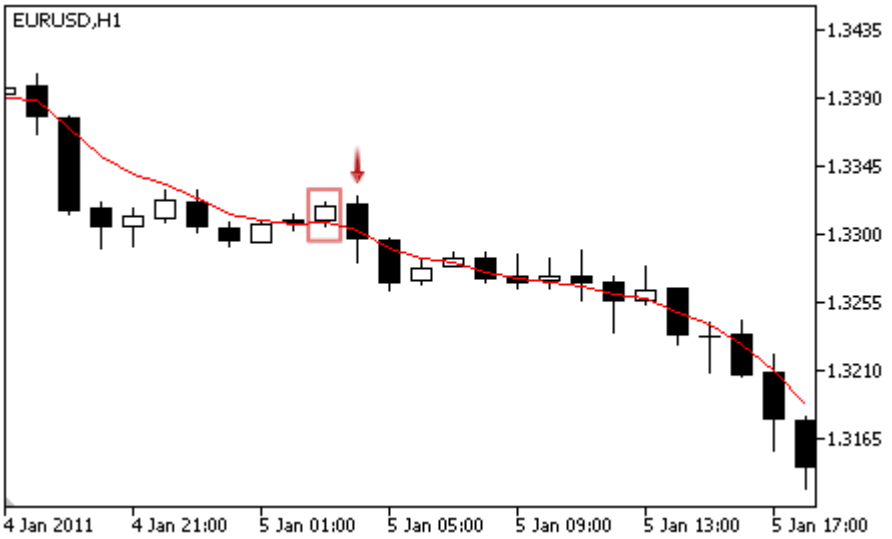
Este módulo se basa en los modelos de mercado del indicador Media Móvil Exponencial Doble. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

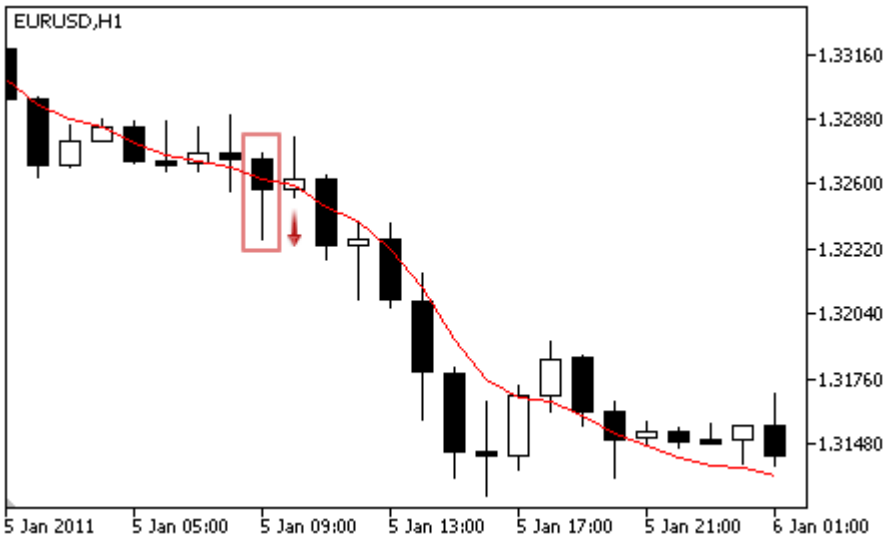
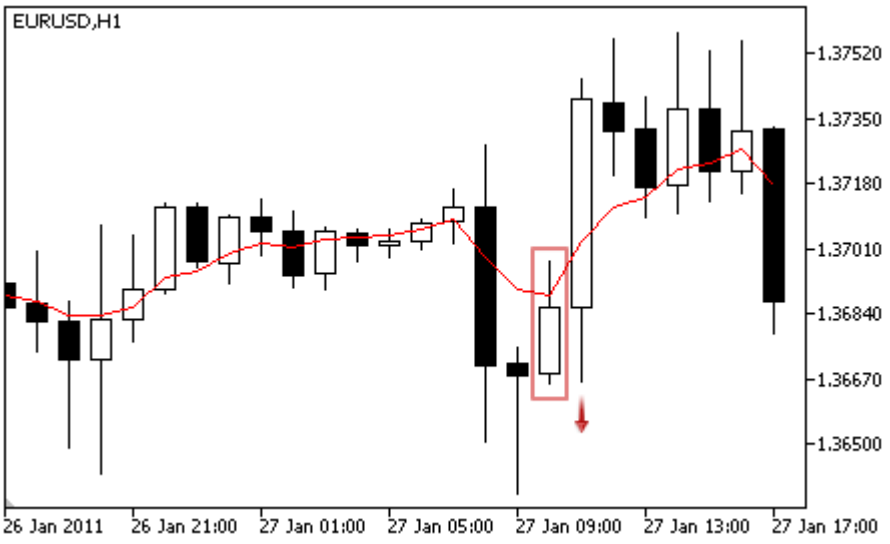
A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• El precio ha cruzado el indicador hacia abajo (el precio de apertura de la barra analizada está por encima del indicador, y el precio de cierre se encuentra por debajo del mismo) y el indicador sube (señal débil).  • **Cruce de la media móvil.** El precio ha cruzado el indicador hacia arriba (el precio de apertura de la barra analizada está por debajo del indicador, y el precio de cierre se encuentra por encima del mismo) y el indicador sube (señal fuerte). 

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	• La sombra inferior de la barra ha cruzado el indicador (los precios de apertura y cierre de la barra analizada están por encima del indicador, y el precio bajo se encuentra por debajo del mismo) y el indicador sube (señal débil). 
De venta	• El precio ha cruzado el indicador hacia arriba (el precio de apertura de la barra analizada está por debajo del indicador, y el precio de cierre se encuentra sobre el mismo) y el indicador cae (señal débil).  • **Cruce de la media móvil.** El precio ha cruzado el indicador hacia abajo (el precio de apertura de la barra analizada está por encima del indicador, y el precio de cierre se encuentra por debajo del mismo) y el indicador cae (señal fuerte).

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • La sombra superior de la barra ha cruzado el indicador (los precios de apertura y cierre de la barra analizada están por debajo del indicador, y el precio alto se encuentra sobre el mismo) y el indicador cae (señal débil). 
No hay objeciones para comprar	El precio está por encima del indicador.
No hay objeciones para vender	El precio está por debajo del indicador.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodMA	Período de promediación del indicador.
Shift	Desplazamiento del indicador a lo largo del eje temporal (en barras).
Method	Método de promediación.
Applied	La serie de precios utilizada para calcular el indicador.

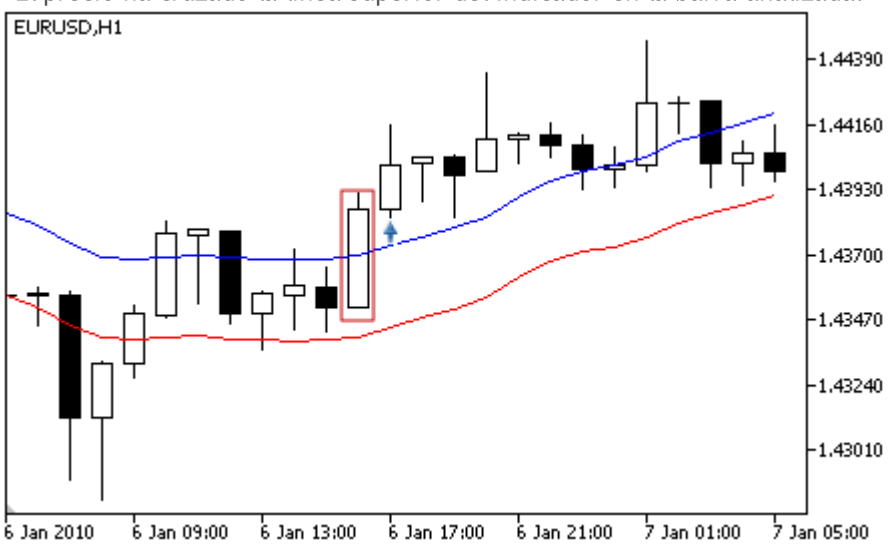
© 2000-2025, MetaQuotes Ltd.

Señales del indicador Envelopes

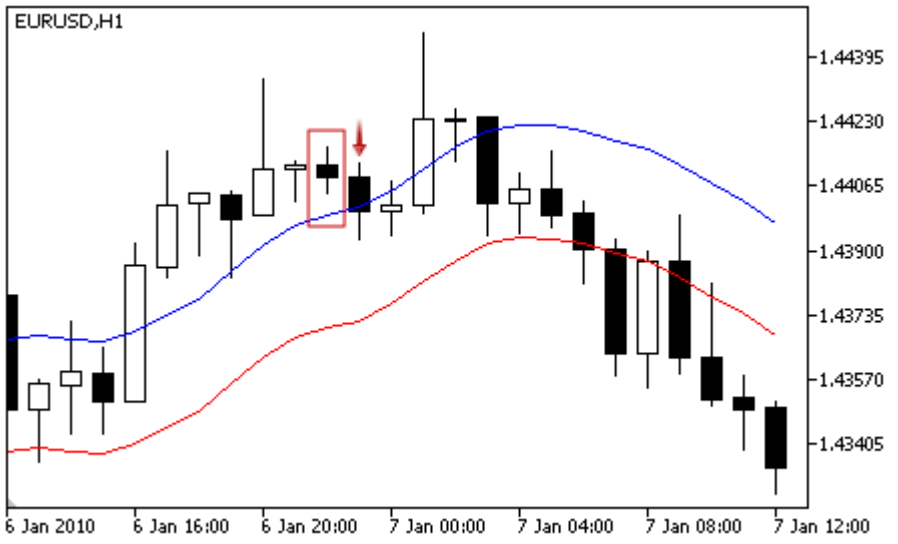
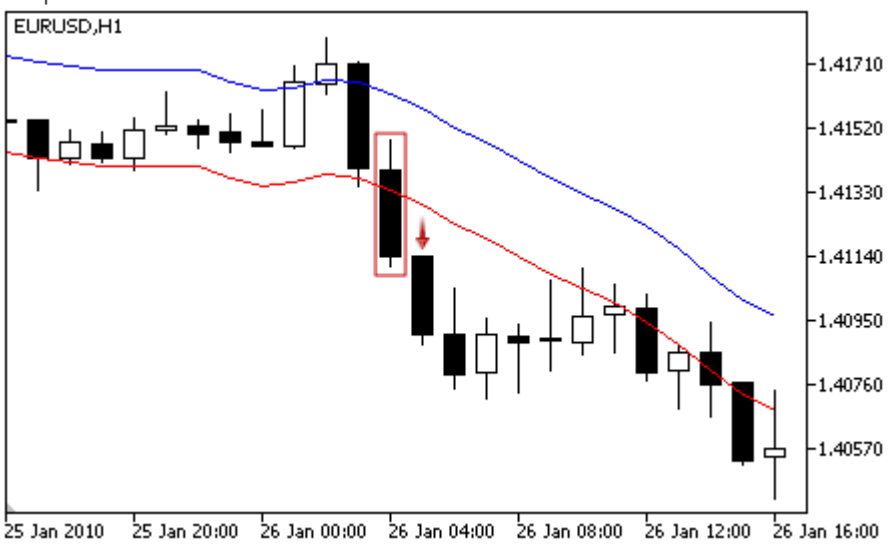
Este módulo de señales se basa en los modelos de mercado del indicador Envelopes. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• El precio está cerca de la línea inferior del indicador en la barra analizada.  • El precio ha cruzado la línea superior del indicador en la barra analizada. 
De venta	• El precio está cerca de la línea superior del indicador en la barra analizada.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • El precio ha cruzado la línea inferior del indicador en la barra analizada. 
No hay objeciones para comprar	Sin señales.
No hay objeciones para vender	Sin señales.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodMA	Periodo de cálculo del indicador.
Shift	Desplazamiento del indicador a lo largo del eje temporal (en barras).
Method	Método de promediación.
Applied	La serie de precios utilizada para calcular el indicador.
Deviation	Desviación de los bordes de la envoltura de la línea de central (MA), en tanto por ciento.

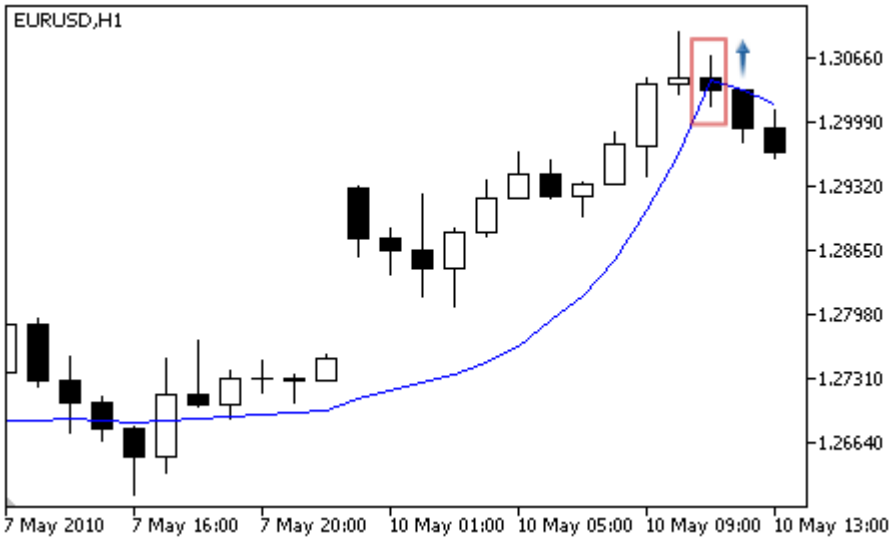
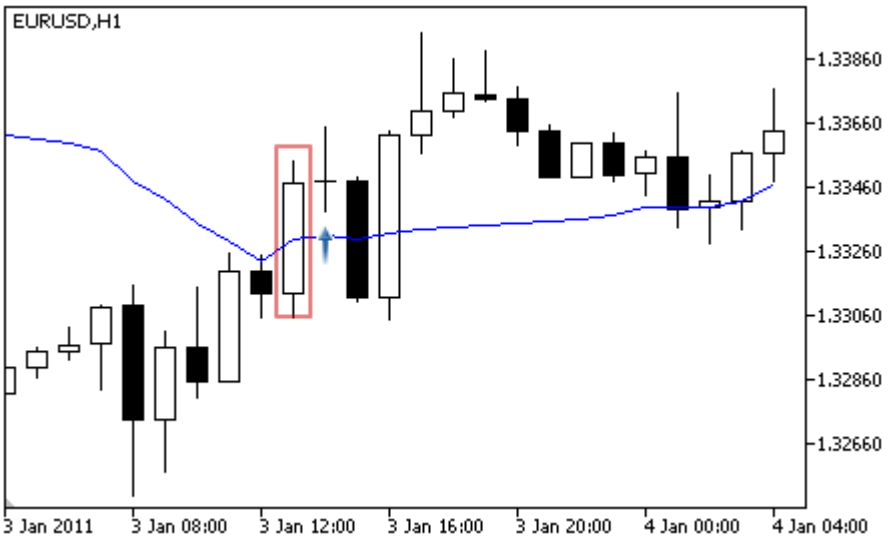
© 2000-2025, MetaQuotes Ltd.

Señales del indicador Media Móvil Adaptativa Fractal

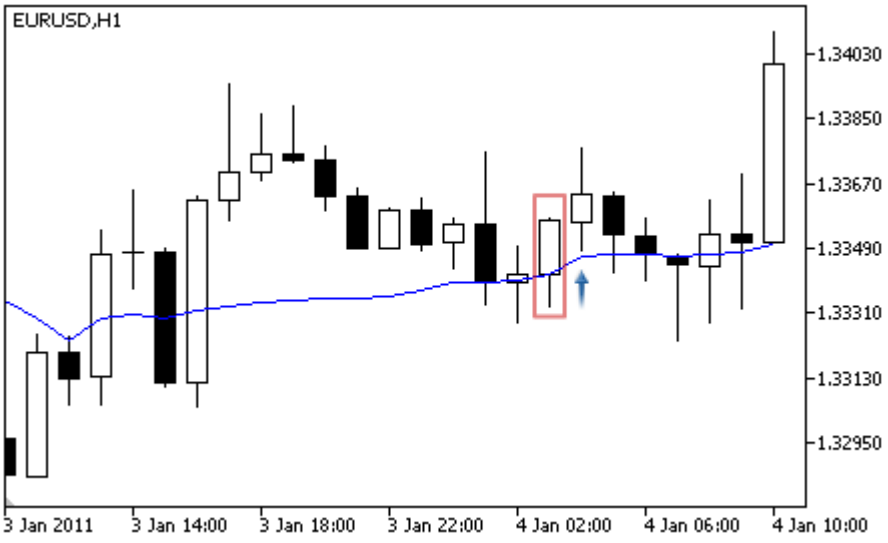
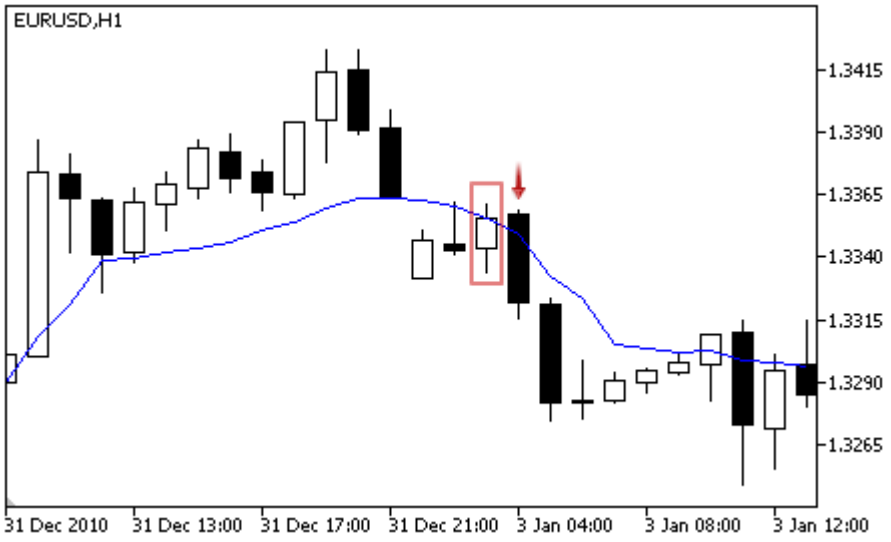
Este módulo de señales se basa en los modelos de mercado del indicador Media Móvil Adaptativa Fractal. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

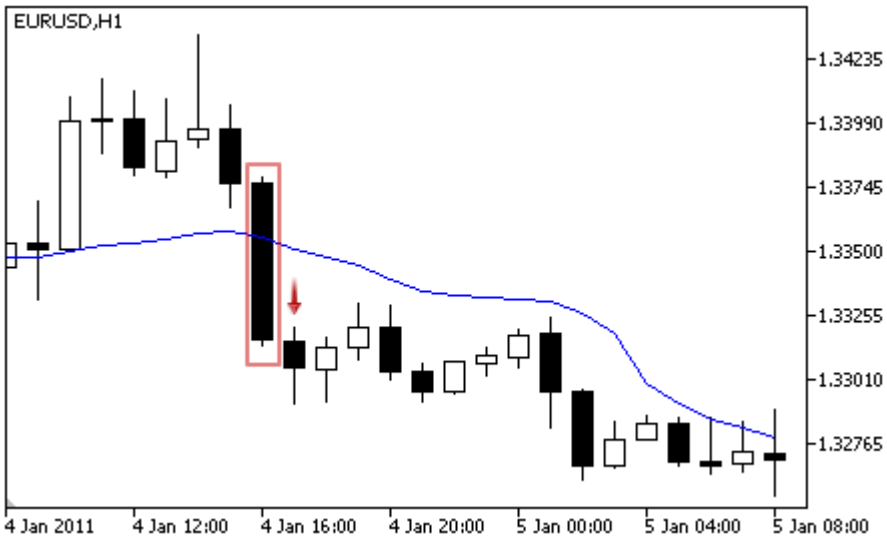
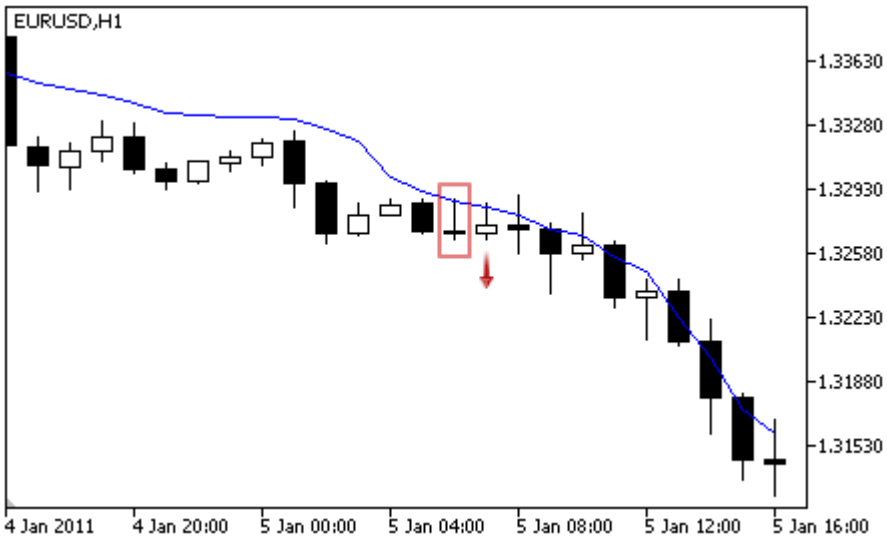
A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• El precio ha cruzado el indicador hacia abajo (el precio de apertura de la barra analizada está por encima del indicador, y el precio de cierre se encuentra por debajo del mismo) y el indicador sube (señal débil).  • **Cruce de la media móvil.** El precio ha cruzado el indicador hacia arriba (el precio de apertura de la barra analizada está por debajo del indicador, y el precio de cierre se encuentra por encima del mismo) y el indicador sube (señal fuerte). 

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	• La sombra inferior de la barra ha cruzado el indicador (los precios de apertura y cierre de la barra analizada están por encima del indicador, y el precio bajo se encuentra por debajo del mismo) y el indicador sube (señal débil). 
De venta	• El precio ha cruzado el indicador hacia arriba (el precio de apertura de la barra analizada está por debajo del indicador, y el precio de cierre se encuentra sobre el mismo) y el indicador cae (señal débil).  • **Cruce de la media móvil.** El precio ha cruzado el indicador hacia abajo (el precio de apertura de la barra analizada está por encima del indicador, y el precio de cierre se encuentra por debajo del mismo) y el indicador cae (señal fuerte).

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • La sombra superior de la barra ha cruzado el indicador (los precios de apertura y cierre de la barra analizada están por debajo del indicador, y el precio alto se encuentra sobre el mismo) y el indicador cae (señal débil). 
No hay objeciones para comprar	El precio está por encima del indicador.
No hay objeciones para vender	El precio está por debajo del indicador.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodMA	Período de promediación del indicador.
Shift	Desplazamiento del indicador a lo largo del eje temporal (en barras).
Method	Método de promediación.
Applied	La serie de precios utilizada para calcular el indicador.

© 2000-2025, MetaQuotes Ltd.

Señales del Filtro de Tiempo Intradía

Este módulo se basa en la suposición de que la eficiencia de los modelos de mercado cambia con el tiempo. Con este módulo, se pueden filtrar las señales recibidas de otros módulos, por horas y días de la semana. Permite aumentar la calidad de las señales generadas porque limita los períodos desfavorables. El mecanismo de toma de decisiones basadas en las señales de los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	Sin señales.
De venta	Sin señales.
No hay objeciones para comprar	La fecha y hora actuales cumplen los parámetros especificados.
No hay objeciones para vender	La fecha y hora actuales cumplen los parámetros especificados.

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
GoodHourOfDay	Número de la única hora del día (de 0 a 23) cuando se activan las señales de trading. Si el valor es -1, las señales se activan durante todo el día.
BadHoursOfDay	El campo bit. Cada bit de este campo corresponde a una hora del día (bit 0 - hora 0, ..., bit 23 - hora 23). Si el valor de un bit es igual a 0, las señales se activan durante la hora correspondiente. Si el valor de un bit es igual a 1, las señales se desactivan durante la hora correspondiente. Los números se representan en binario y se utilizan como máscara de bits. Las horas desactivadas tienen mayor prioridad que las activadas.
GoodDayOfWeek	Número del único día de la semana (de 0 a 6, donde 0 es el domingo), cuando se activan las señales de trading. Si el valor es -1, las señales se activan cualquier día.
BadDaysOfWeek	El campo bit. Cada bit de este campo corresponde a un día de semana (bit 0 - domingo, ..., bit 6 - sábado). Si el valor de un bit es igual a 0, las señales de trading se activan durante el día que corresponda. Si el valor de un bit es igual a 1, las señales de trading se desactivan durante

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
	el día correspondiente. Los números se representan en binario y se utilizan como máscara de bits. Los días desactivados tienen mayor prioridad que los habilitados.

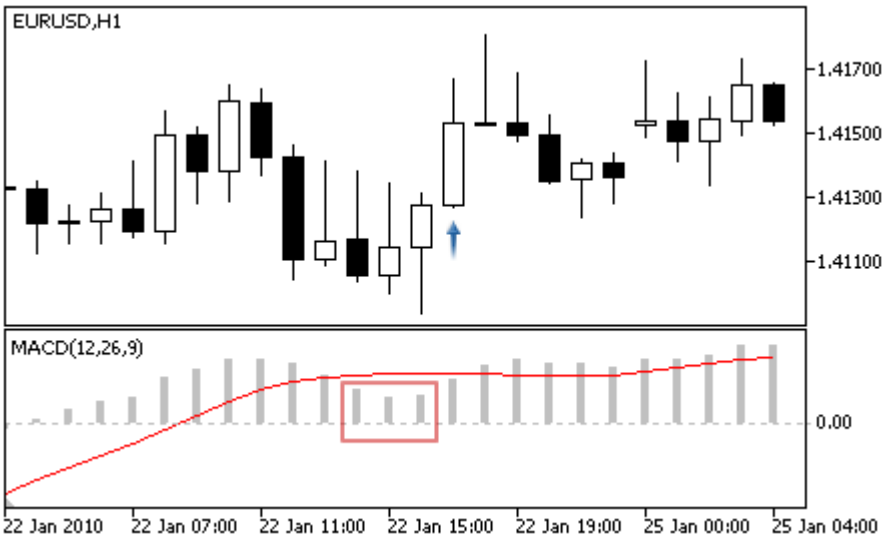
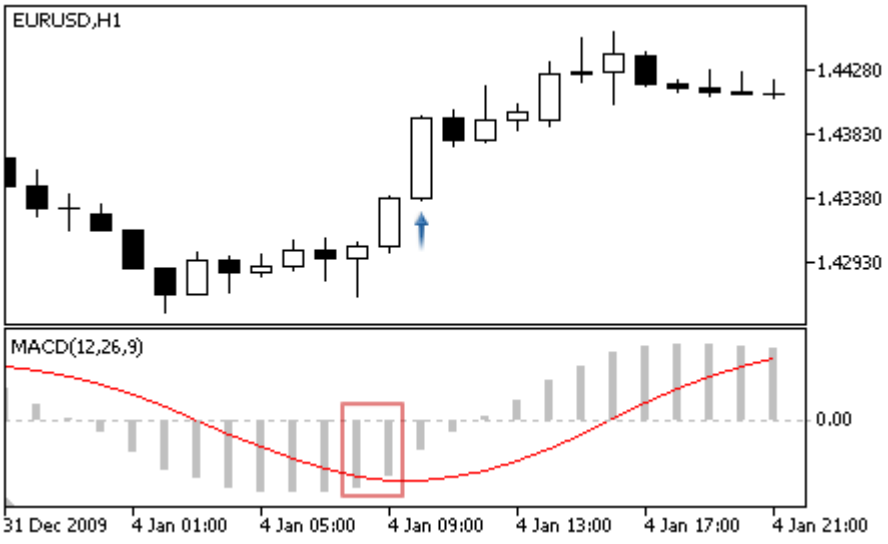
© 2000-2025, MetaQuotes Ltd.

Señales del oscilador MACD

Este módulo de señales se basa en los modelos de mercado del oscilador MACD. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

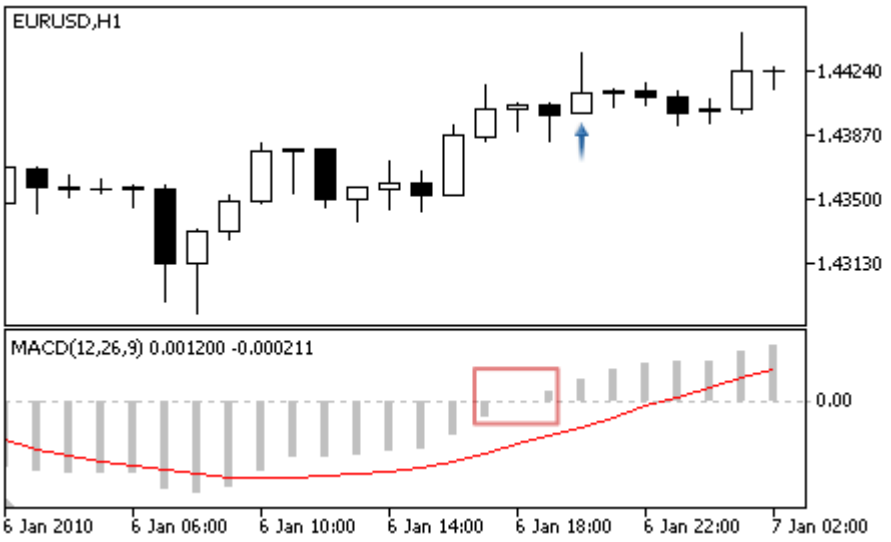
Condiciones para la generación de las señales

A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

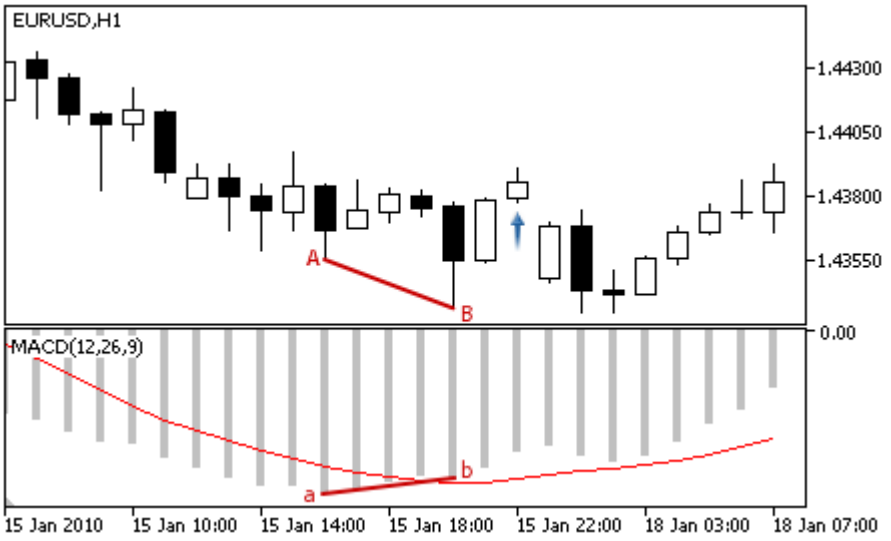
Tipo de señal	Descripción de las condiciones
De compra	• **Inversión** — el oscilador se gira hacia arriba, subiendo en el barra analizada y cayendo en la anterior.  • **Cruce de la línea principal con la línea de señal** — la línea principal está por encima de la línea de señal en la barra analizada, y por debajo de la misma en la barra anterior.  • **Cruce del nivel cero** — la línea principal está por encima del nivel cero en la barra analizada, y por debajo del nivel cero en la barra anterior.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones

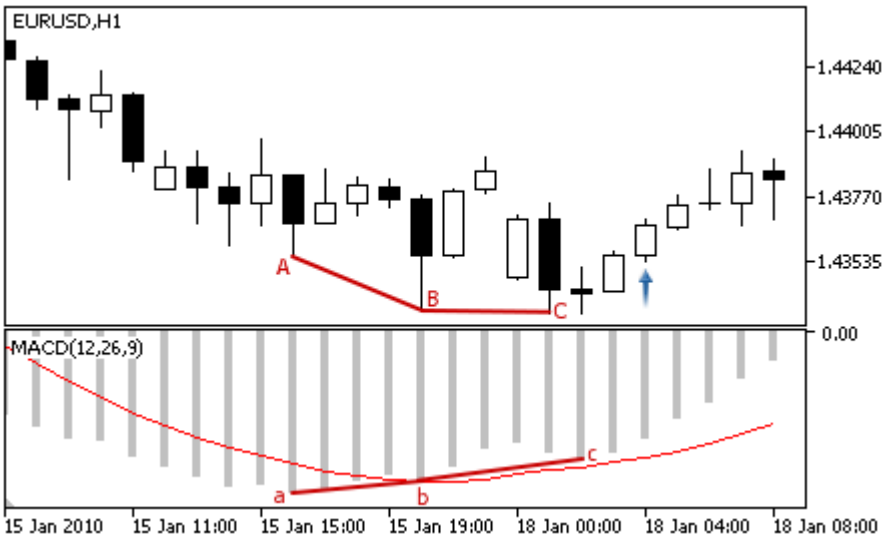
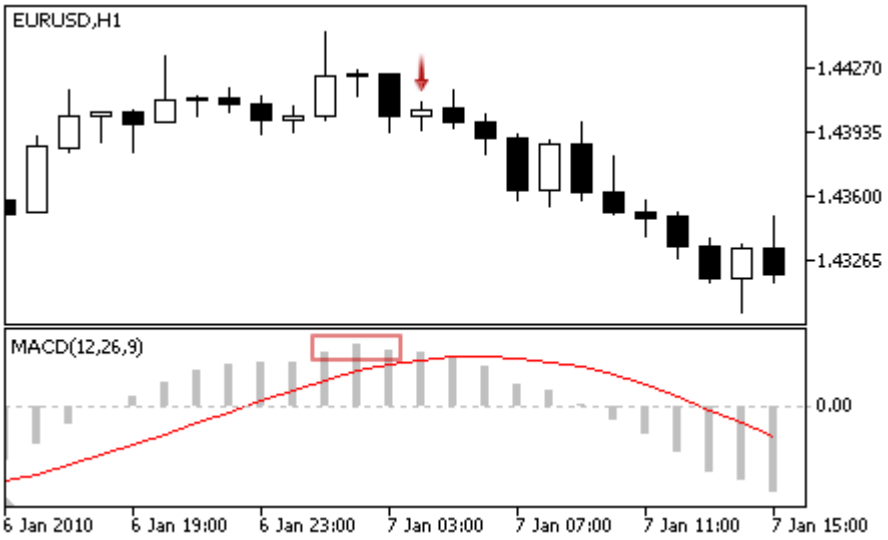


- **Divergencia** — el primer pico inferior analizado por el oscilador es mayor que el anterior, y el pico inferior del precio correspondiente es más bajo que el anterior.



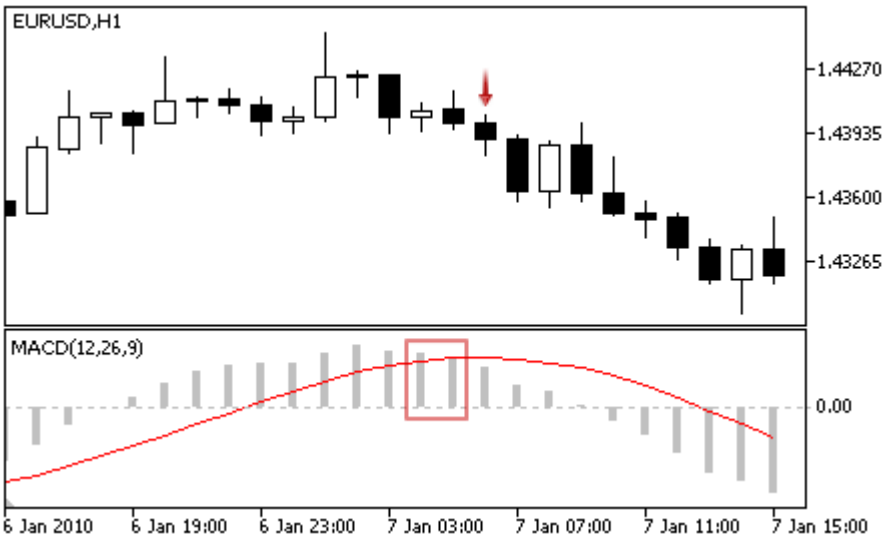
- **Divergencia doble** — el oscilador forma tres picos inferiores consecutivos, cada uno mayor que el anterior; y el precio forma los tres picos inferiores correspondientes, siendo cada uno de ellos menor que el anterior.

© 2000-2025, MetaQuotes Ltd.

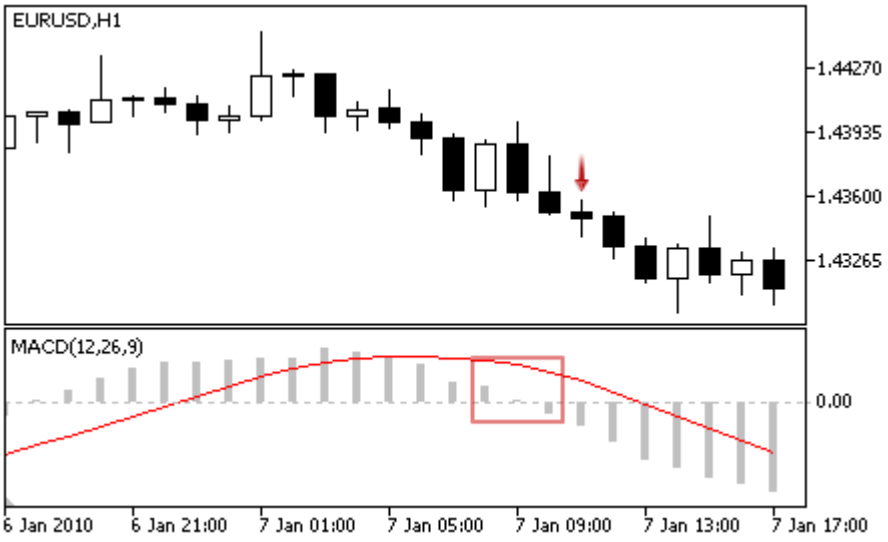
Tipo de señal	Descripción de las condiciones
	
De venta	• **Inversión** — el oscilador se gira hacia abajo, cayendo en la barra analizada y subiendo en la barra anterior.  • **Cruce de la línea principal con la línea de señal** — la línea principal está por debajo de la línea de señal en la barra analizada, y por encima de la misma en la barra anterior.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones

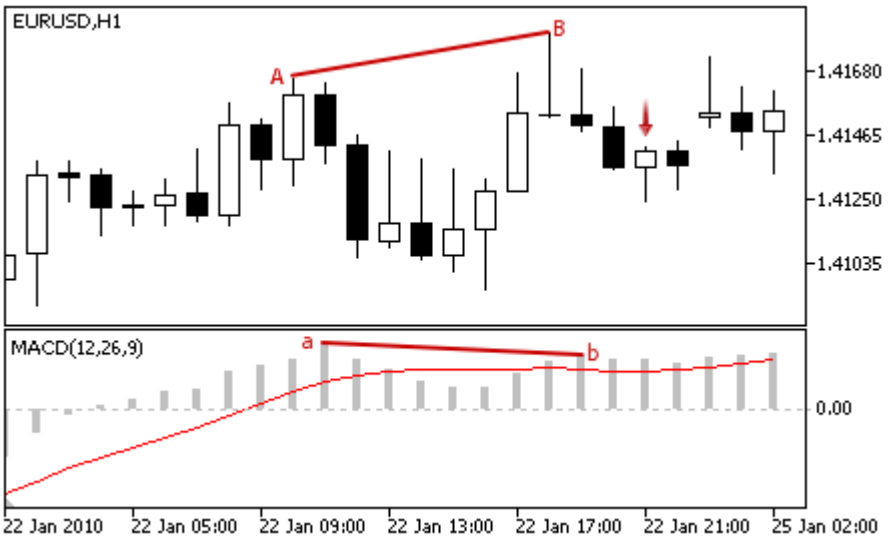
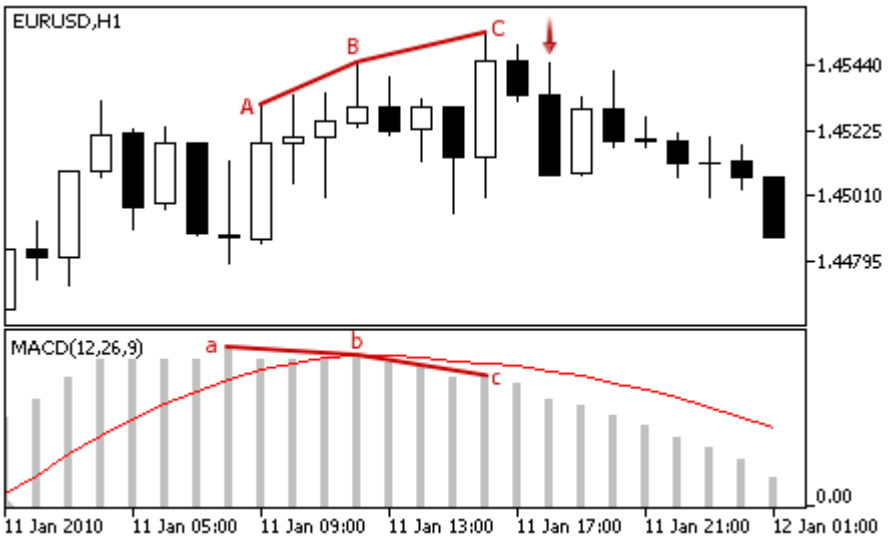


- **Cruce del nivel cero** — la línea principal está por debajo del nivel cero en la barra analizada, y por encima del nivel cero en la barra anterior.



- **Divergencia** — el primer pico analizado del oscilador es menor que el anterior, y el pico del precio correspondiente es más alto que el pico anterior.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • **Divergencia doble** — el oscilador forma tres picos consecutivos, cada uno menor que el anterior; y el precio forma tres picos correspondientes, siendo cada uno de ellos mayor que el anterior. 
No hay objeciones para comprar	El valor del oscilador crece en la barra analizada.
No hay objeciones para vender	El valor del oscilador cae en la barra analizada.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodFast	Periodo de cálculo de la EMA rápida.
PeriodSlow	Periodo de cálculo de la EMA lenta.
PeriodSignal	Periodo de suavizado.
Applied	La serie de precios utilizada en el cálculo del oscilador.

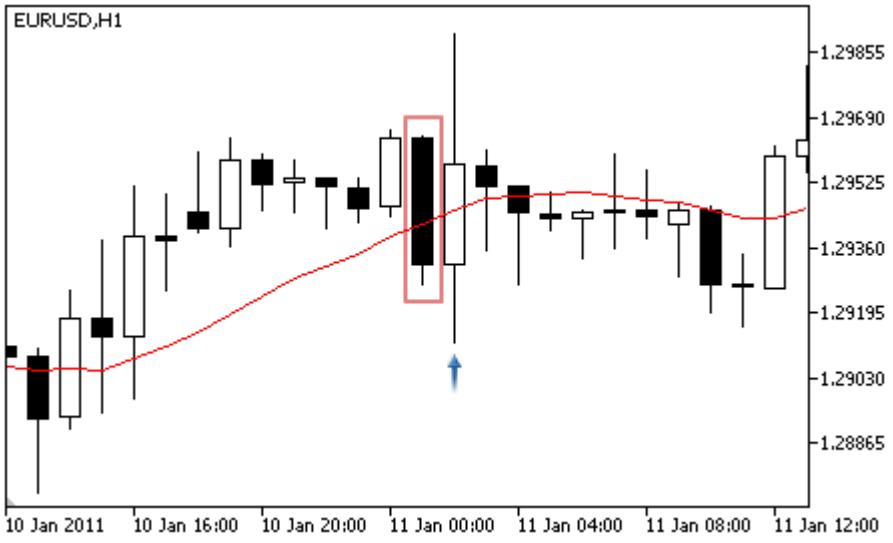
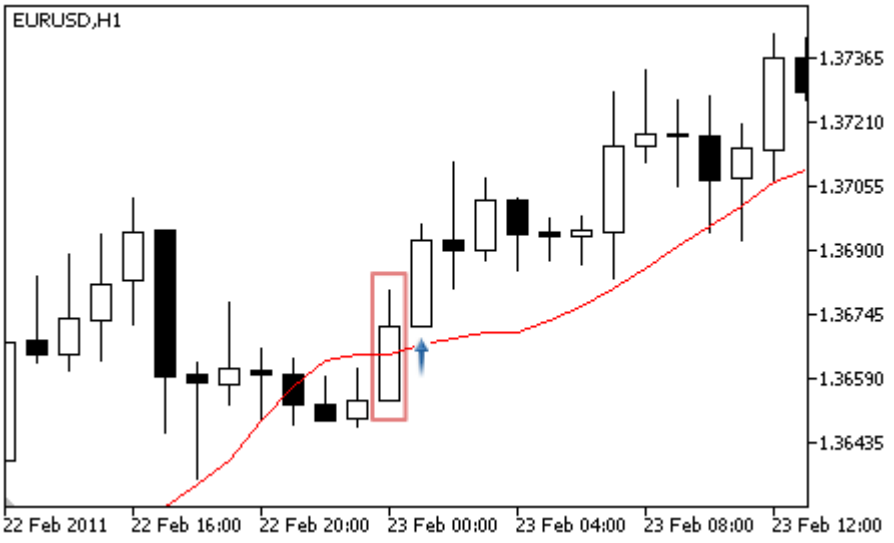
© 2000-2025, MetaQuotes Ltd.

Señales del indicador Media Móvil

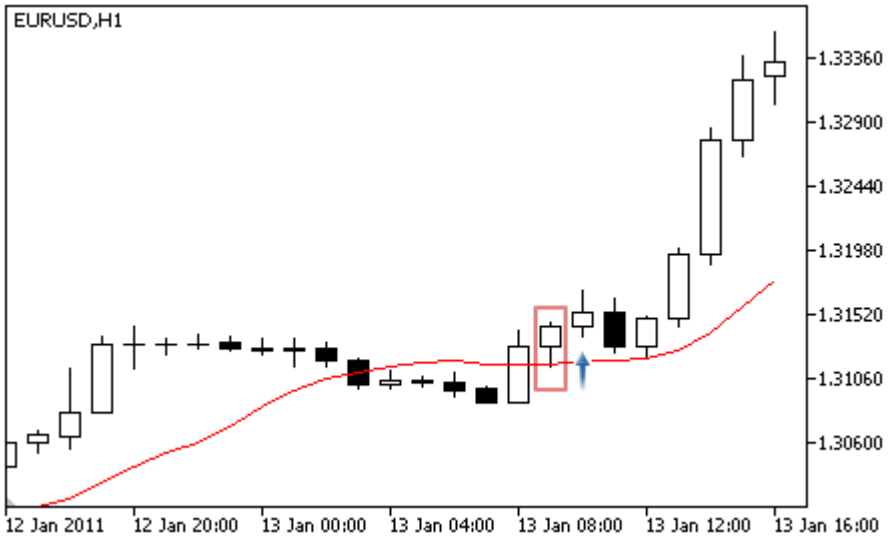
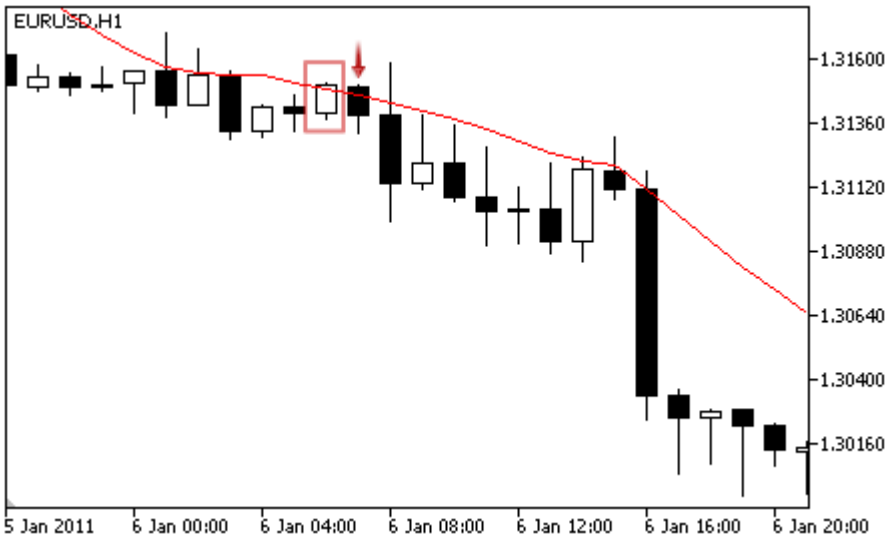
Este módulo de señales se basa en los modelos de mercado del indicador Media Móvil. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

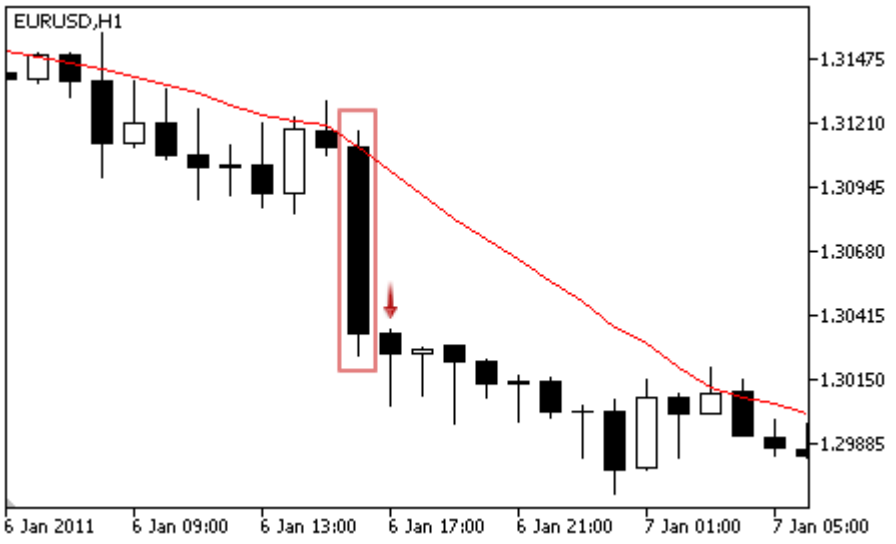
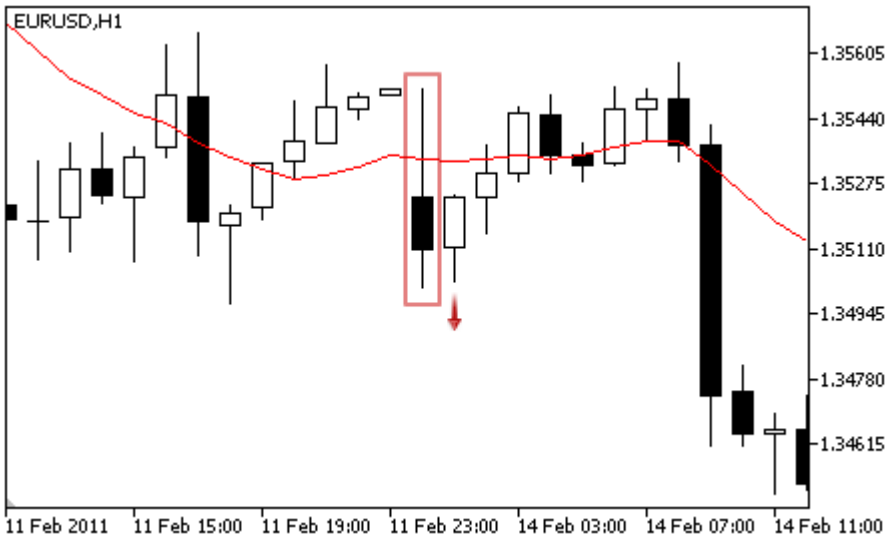
A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• El precio ha cruzado el indicador hacia abajo (el precio de apertura de la barra analizada está por encima del indicador, y el precio de cierre se encuentra por debajo del mismo) y el indicador sube (señal débil).  • **Cruce de la media móvil.** El precio ha cruzado el indicador hacia arriba (el precio de apertura de la barra analizada está por debajo del indicador, y el precio de cierre se encuentra por encima del mismo) y el indicador sube (señal fuerte). 

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	• La sombra inferior de la barra ha cruzado el indicador (los precios de apertura y cierre de la barra analizada están por encima del indicador, y el precio bajo se encuentra por debajo del mismo) y el indicador sube (señal débil). 
De venta	• El precio ha cruzado el indicador hacia arriba (el precio de apertura de la barra analizada está por debajo del indicador, y el precio de cierre se encuentra sobre el mismo) y el indicador cae (señal débil).  • **Cruce de la media móvil.** El precio ha cruzado el indicador hacia abajo (el precio de apertura de la barra analizada está por encima del indicador, y el precio de cierre se encuentra por debajo del mismo) y el indicador cae (señal fuerte).

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • La sombra superior de la barra ha cruzado el indicador (los precios de apertura y cierre de la barra analizada están por debajo del indicador, y el precio alto se encuentra sobre el mismo) y el indicador cae (señal débil). 
No hay objeciones para comprar	El precio está por encima del indicador.
No hay objeciones para vender	El precio está por debajo del indicador.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodMA	Período de promediación del indicador.
Shift	Desplazamiento del indicador a lo largo del eje temporal (en barras).
Method	Método de promediación.
Applied	La serie de precios utilizada para calcular el indicador.

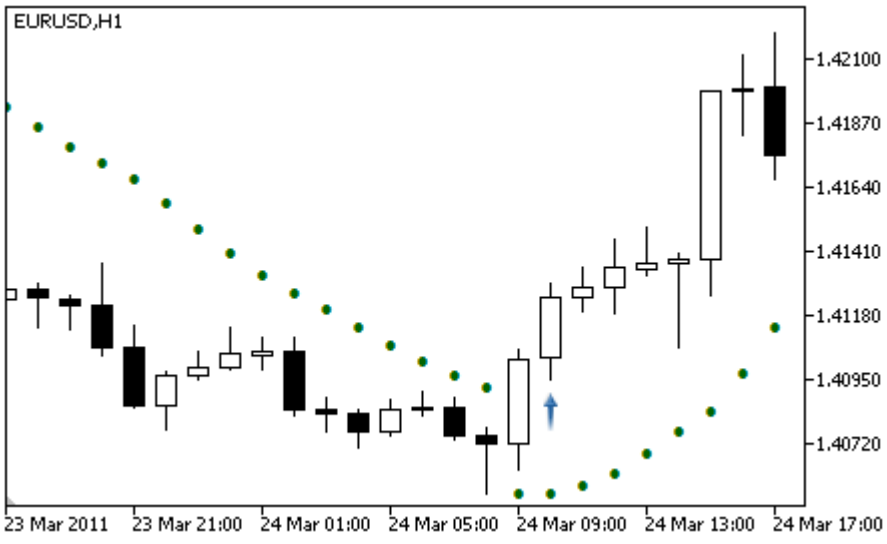
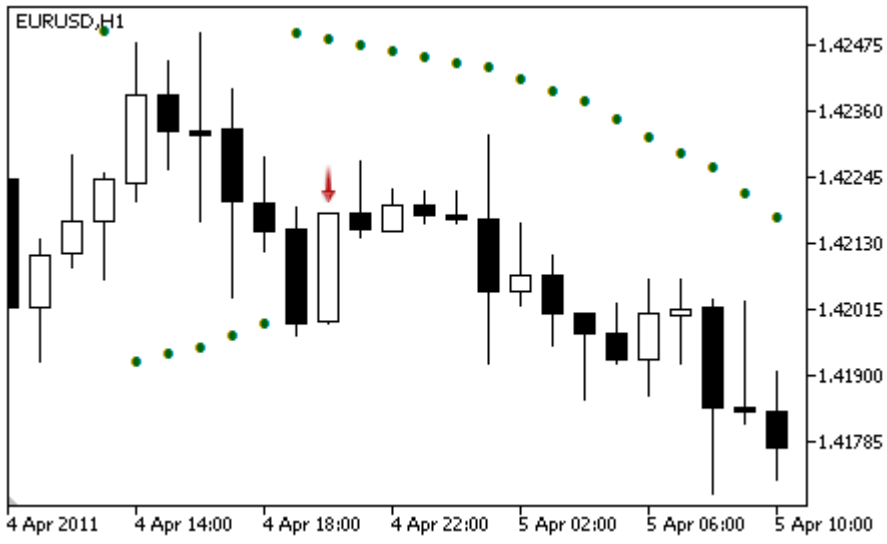
© 2000-2025, MetaQuotes Ltd.

Señales del indicador SAR Parabólico

Este módulo de señales se basa en los modelos de mercado del indicador SAR Parabólico. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	**Inversión** — el indicador está por debajo del precio en la barra analizada, y por encima del mismo en la barra anterior. 
De venta	**Inversión** — el indicador está por encima del precio en la barra analizada, y por debajo del mismo en la barra anterior. 
No hay objeciones para comprar	El precio está por encima del indicador.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
No hay objeciones para vender	El precio está por debajo del indicador.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
Step	El incremento de la velocidad del indicador.
Maximum	Tasa máxima de la velocidad de convergencia del indicador con el precio.

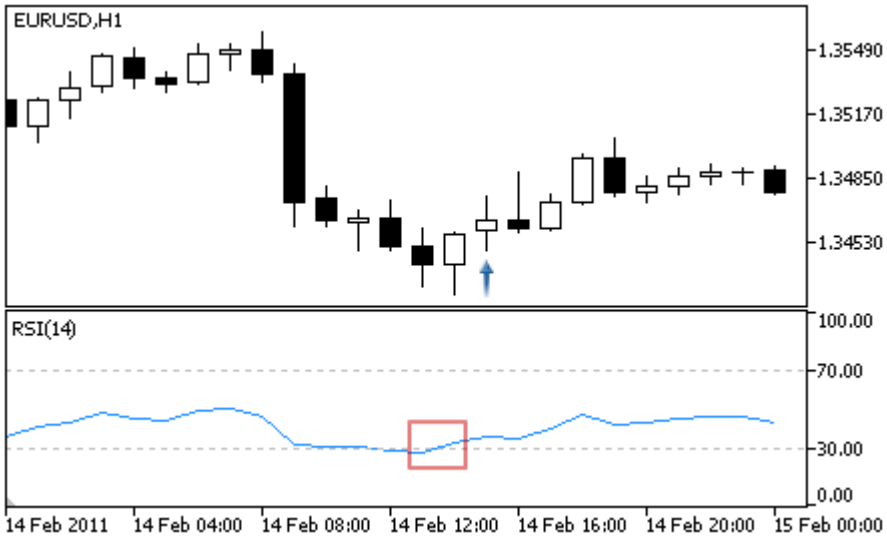
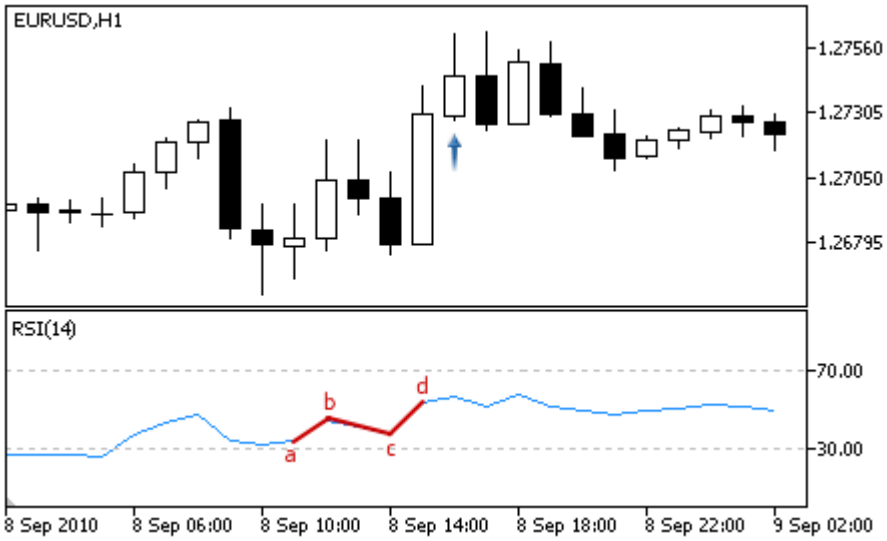
© 2000-2025, MetaQuotes Ltd.

Señales del oscilador Índice de Fuerza Relativa

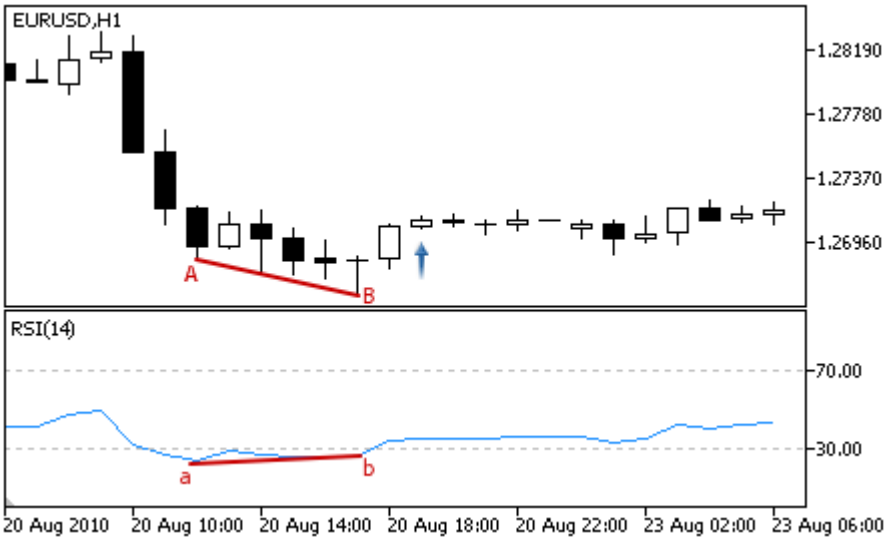
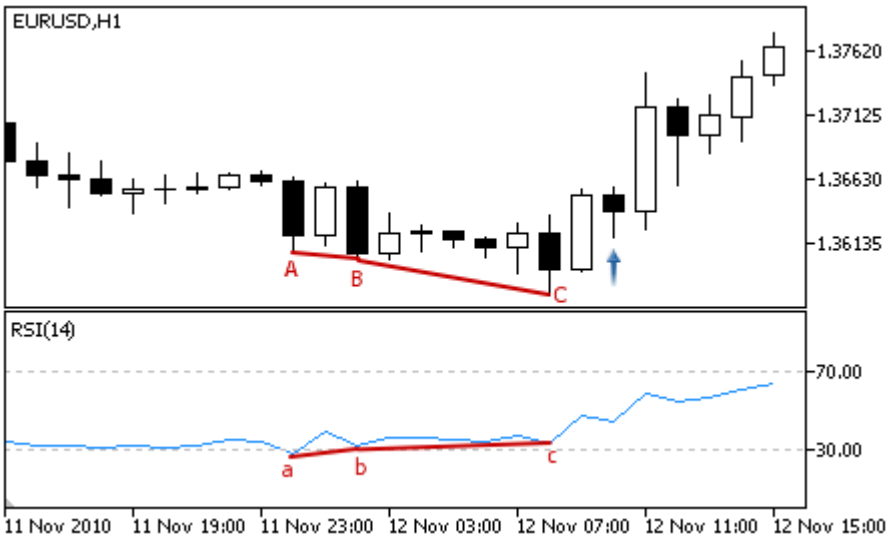
Este módulo de señales se basa en los modelos de mercado del oscilador Índice de Fuerza Relativa. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

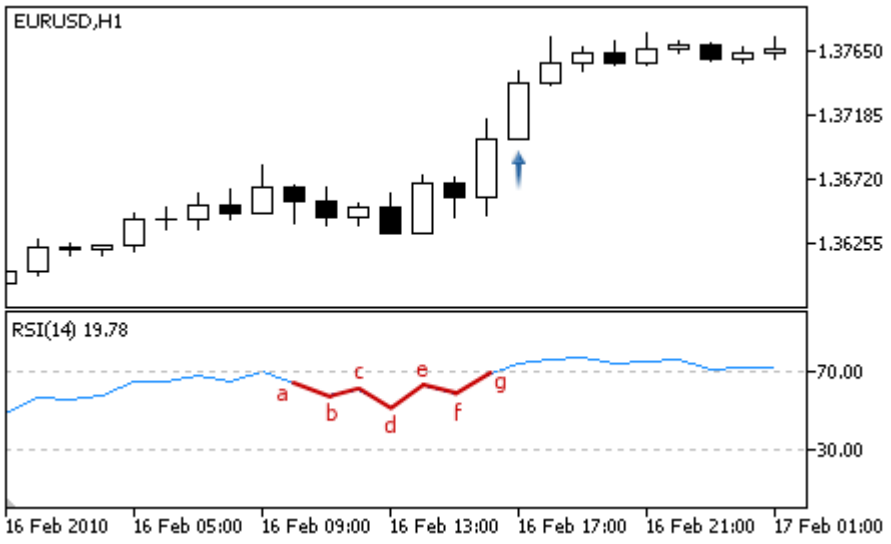
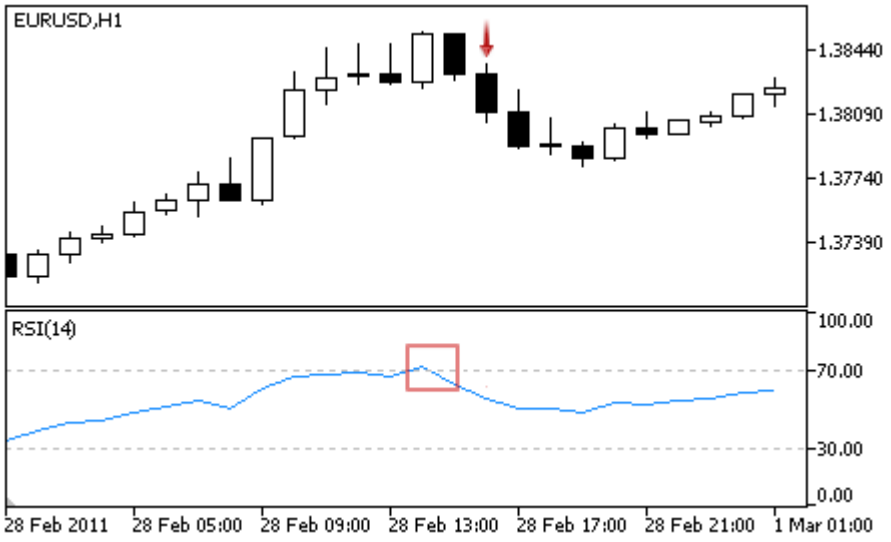
A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• **Inversión por detrás del nivel de sobreventa** — el oscilador se gira hacia arriba y su valor en la barra analizada está detrás del nivel de sobreventa (el valor predeterminado es 30).  • **Swing fallido** — el oscilador se eleva por encima del pico anterior en la barra analizada. 

© 2000-2025, MetaQuotes Ltd.

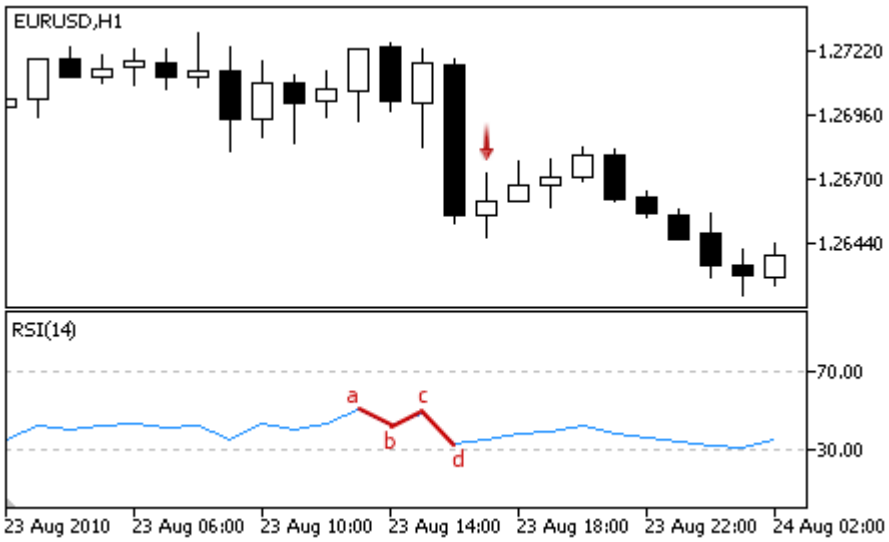
Tipo de señal	Descripción de las condiciones
	• **Divergencia** — el primer pico inferior analizado por el oscilador es menor que el anterior, y el suelo correspondiente del precio es menor que el anterior.  • **Divergencia doble** — el oscilador forma tres picos inferiores consecutivos, cada uno mayor que el anterior; y el precio forma los tres picos inferiores correspondientes, siendo cada uno de ellos menor que el anterior.  • **Cabeza/Hombros** — el oscilador forma tres picos inferiores consecutivos, y el del medio es más bajo que los otros dos.

© 2000-2025, MetaQuotes Ltd.

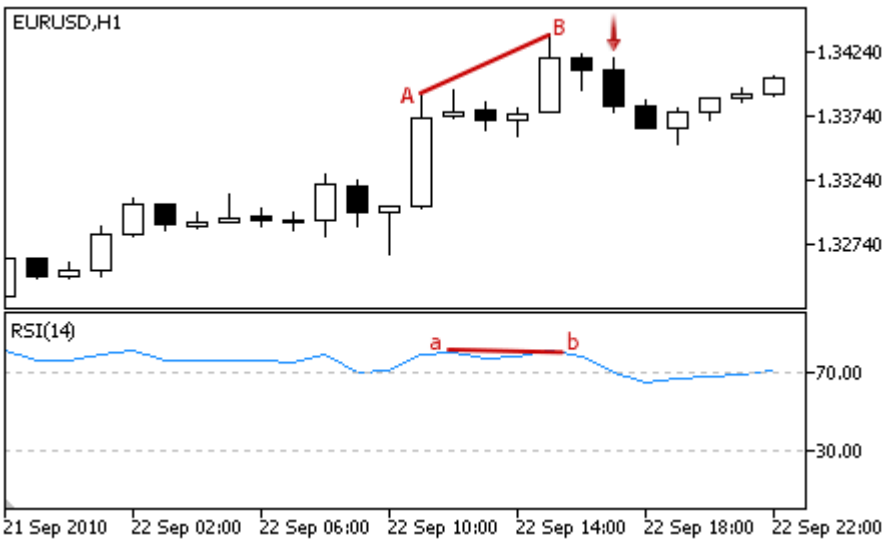
Tipo de señal	Descripción de las condiciones
	
De venta	• **Inversión por detrás del nivel de sobrecompra** — el oscilador se gira hacia abajo y su valor en la barra analizada está detrás del nivel de sobrecompra (el valor predeterminado es 70).  • **Swing fallido** — el oscilador cae por debajo del pico inferior anterior en la barra analizada.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones

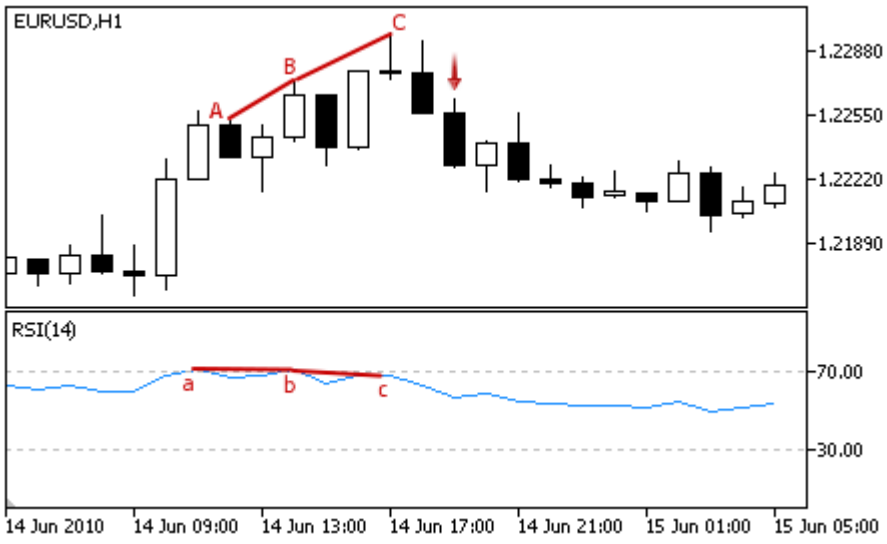
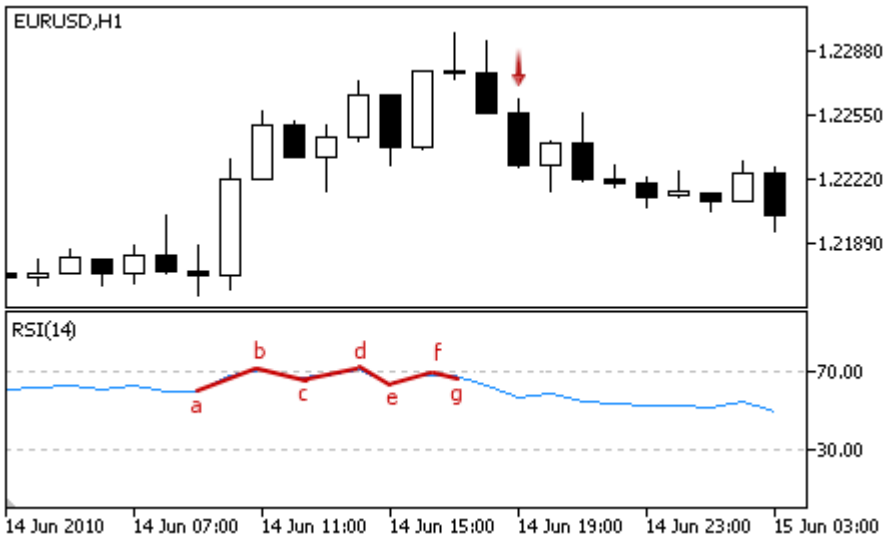


- **Divergencia** — el primer pico analizado del oscilador es menor que el anterior, y el pico del precio correspondiente es más alto que el pico anterior.



- **Divergencia doble** — el oscilador forma tres picos consecutivos, cada uno menor que el anterior; y el precio forma tres picos correspondientes, siendo cada uno de ellos mayor que el anterior.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • **Cabeza/Hombros** — el oscilador forma tres picos consecutivos, y el del medio es más alto que los otros dos. 
No hay objeciones para comprar	El valor del oscilador crece en la barra analizada.
No hay objeciones para vender	El valor del oscilador cae en la barra analizada.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodRSI	Período de cálculo del oscilador.
Applied	La serie de precios utilizada en el cálculo del oscilador.

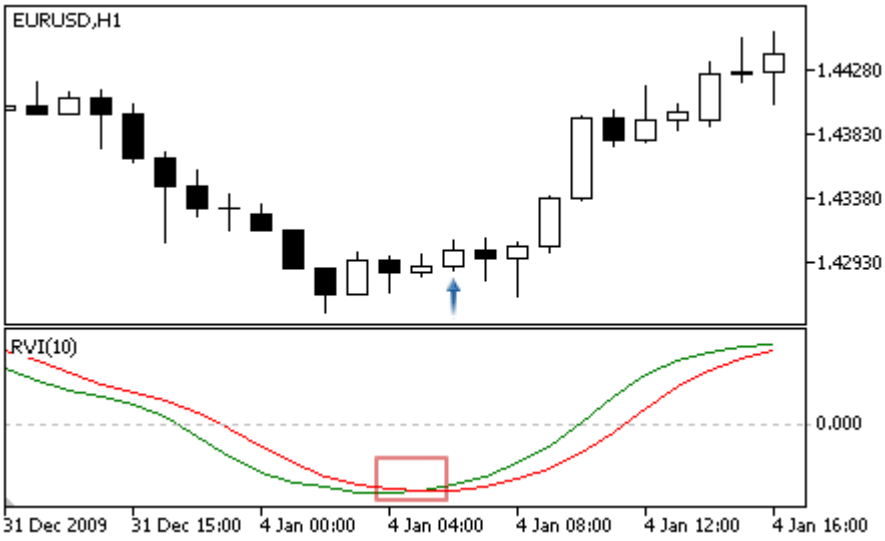
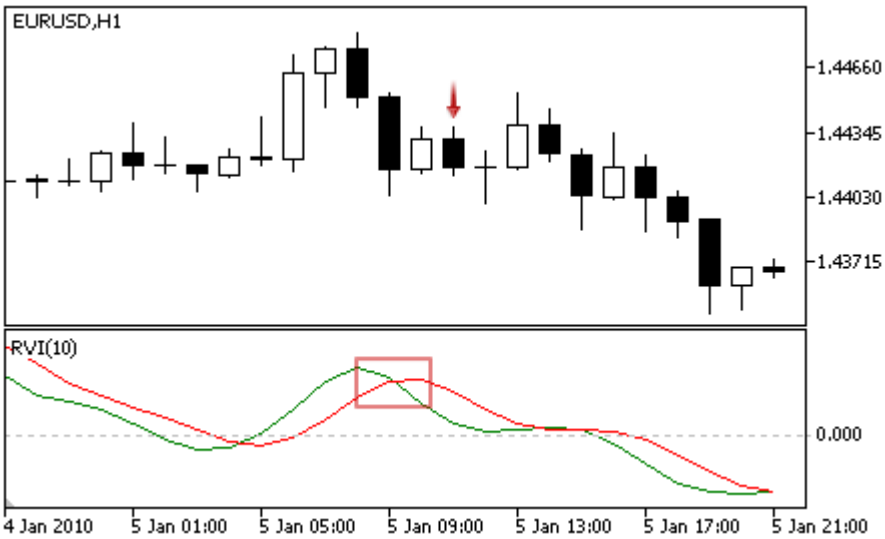
© 2000-2025, MetaQuotes Ltd.

Señales del oscilador Índice de Vigor Relativo

Este módulo de señales se basa en los modelos de mercado del oscilador Índice de Vigor Relativo. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	**Cruce entre la línea principal y la línea de señal** — la línea principal está por encima de la línea de señal en la barra analizada, y por debajo de la misma en la barra anterior. 
De venta	**Cruce entre la línea principal y la línea de señal** — la línea principal está por debajo de la línea de señal en la barra analizada, y por encima de la misma en la barra anterior. 

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
No hay objeciones para comprar	El valor del oscilador crece en la barra analizada.
No hay objeciones para vender	El valor del oscilador cae en la barra analizada.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodRVI	Período de cálculo del oscilador.

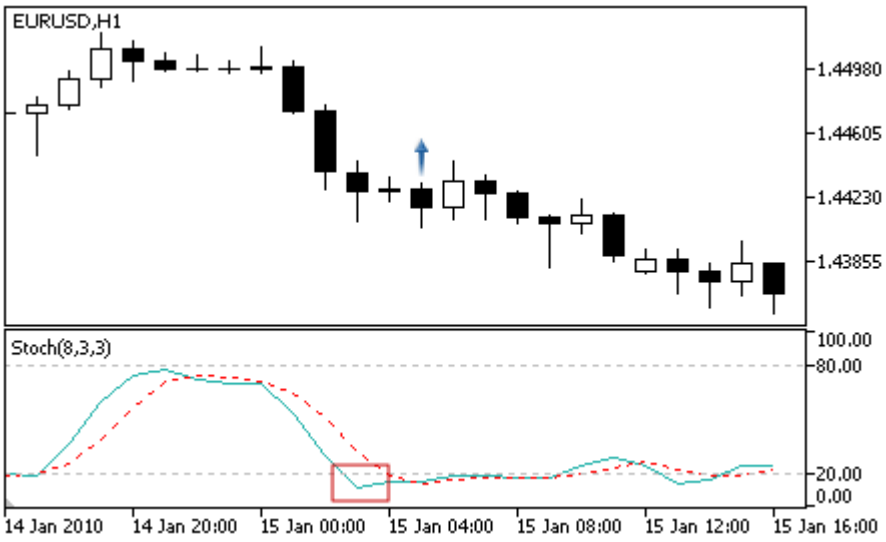
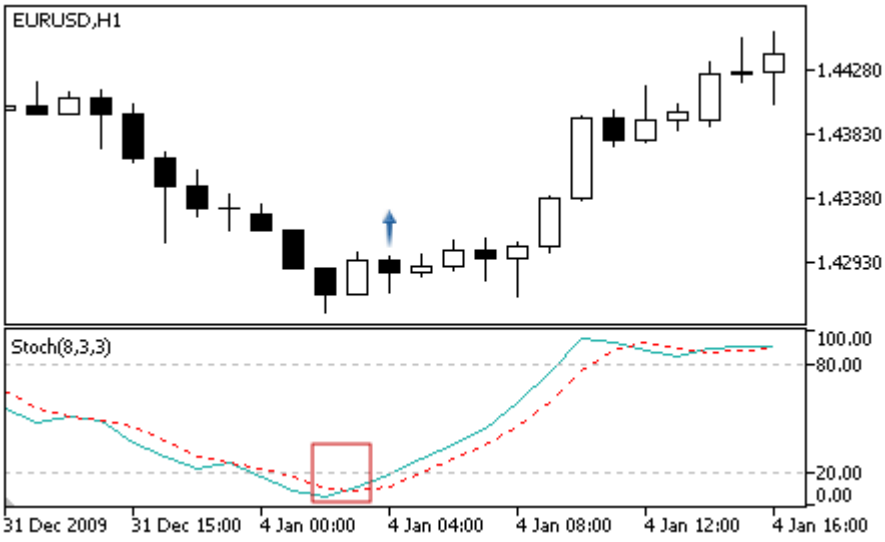
© 2000-2025, MetaQuotes Ltd.

Señales del oscilador Estocástico

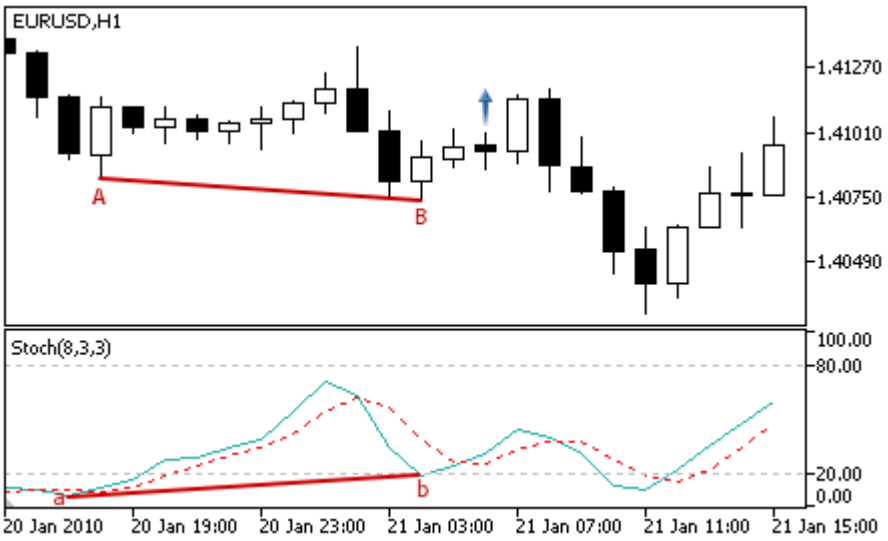
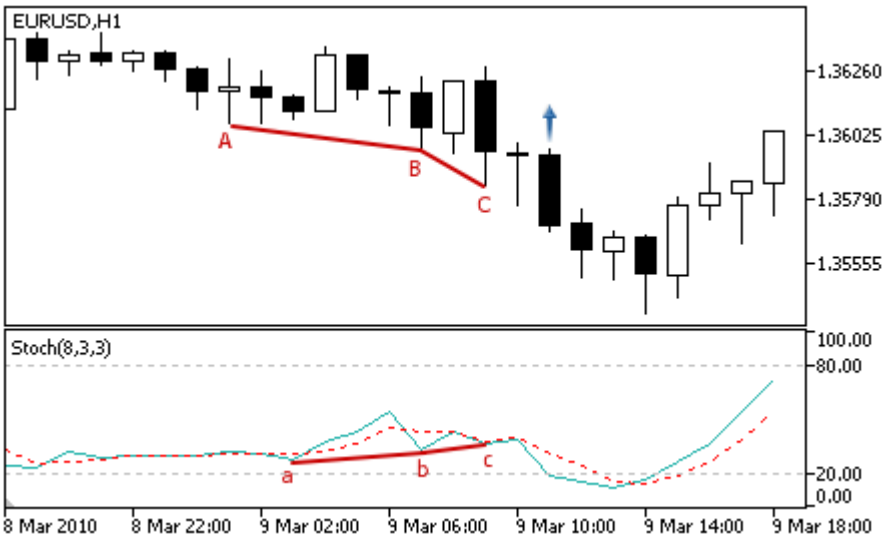
Este módulo de señales se basa en los modelos de mercado del oscilador Estocástico. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

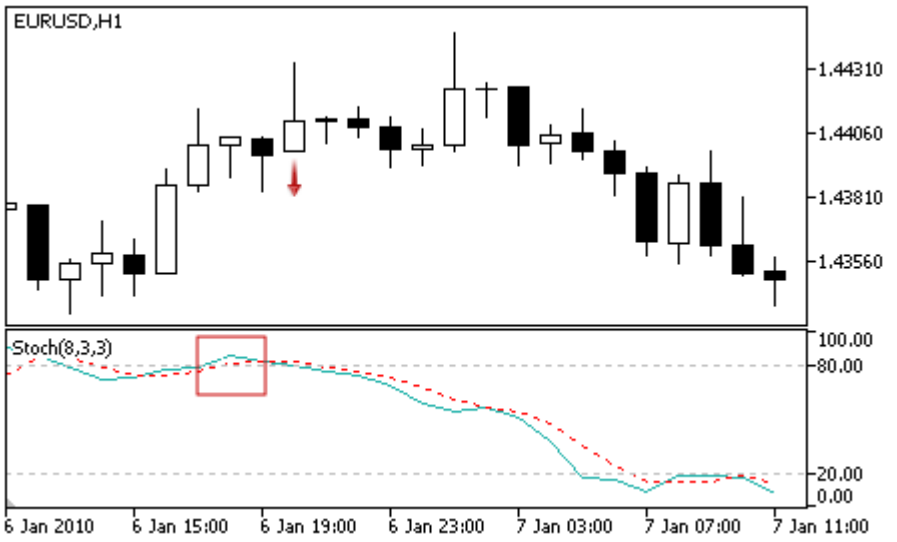
Tipo de señal	Descripción de las condiciones
De compra	• **Inversión** — el oscilador se gira hacia arriba, subiendo en el barra analizada y cayendo en la anterior.  • **Cruce entre la línea principal y la línea de señal** — la línea principal está por encima de la línea de señal en la barra analizada, y por debajo de la misma en la barra anterior. 

© 2000-2025, MetaQuotes Ltd.

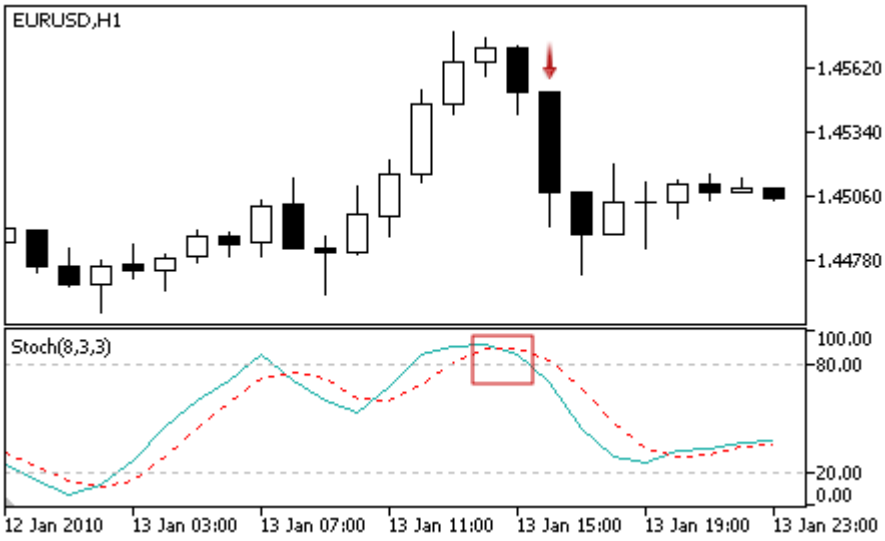
Tipo de señal	Descripción de las condiciones
	• **Divergencia** — el primer pico inferior analizado por el oscilador es mayor que el anterior, y el pico inferior del precio correspondiente es más bajo que el anterior.  • **Divergencia doble** — el oscilador forma tres picos inferiores consecutivos, cada uno mayor que el anterior; y el precio forma los tres picos inferiores correspondientes, siendo cada uno de ellos menor que el anterior. 
De venta	• **Inversión** — el oscilador se gira hacia abajo, cayendo en la barra analizada y subiendo en la barra anterior.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones

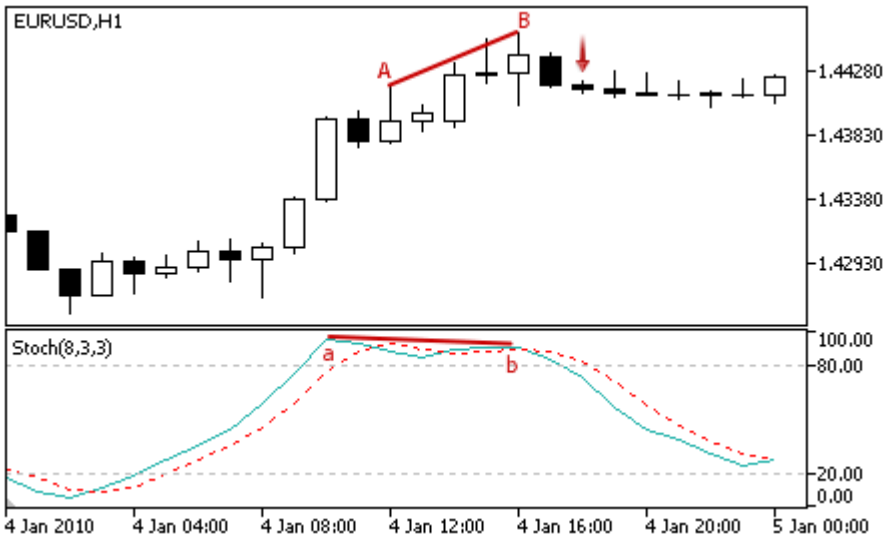
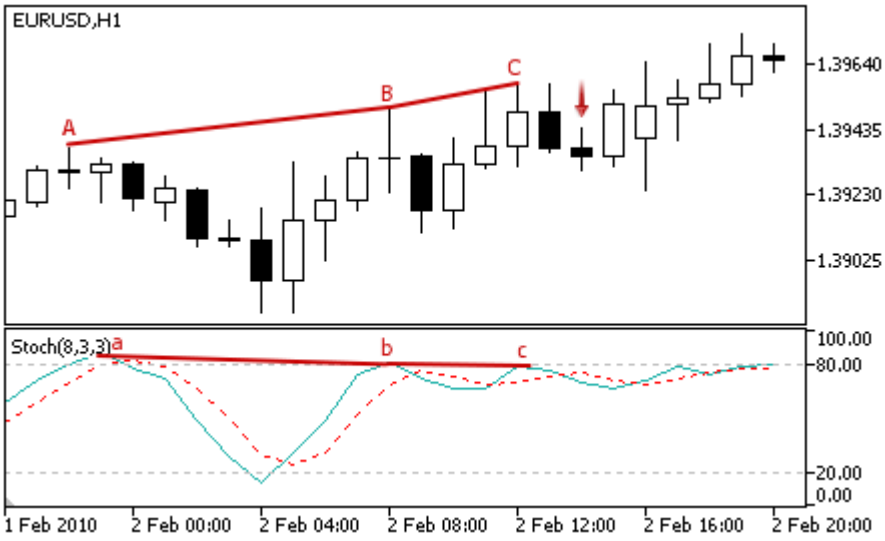


- **Cruce entre la línea principal y la línea de señal** — la línea principal está por debajo de la línea de señal en la barra analizada, y por encima de la misma en la barra anterior.



- **Divergencia** — el primer pico analizado del oscilador es menor que el anterior, y el pico del precio correspondiente es más alto que el pico anterior.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • **Divergencia doble** — el oscilador forma tres picos consecutivos, cada uno menor que el anterior; y el precio forma tres picos correspondientes, siendo cada uno de ellos mayor que el anterior. 
No hay objeciones para comprar	El valor del oscilador crece en la barra analizada.
No hay objeciones para vender	El valor del oscilador cae en la barra analizada.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodK	Período de cálculo de la línea principal del oscilador.
PeriodD	Período de cálculo de la línea principal del oscilador.
PeriodSlow	Período de desaceleración.
Applied	La serie de precios utilizada en el cálculo del oscilador.

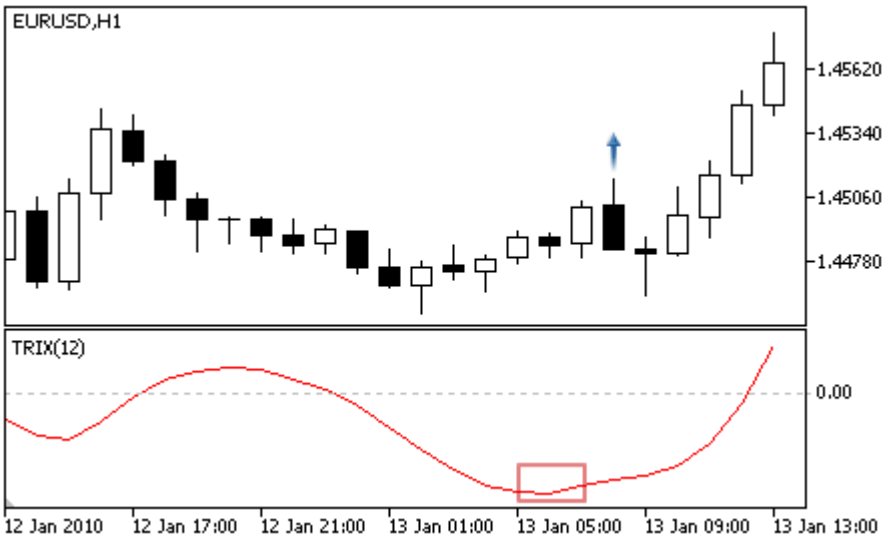
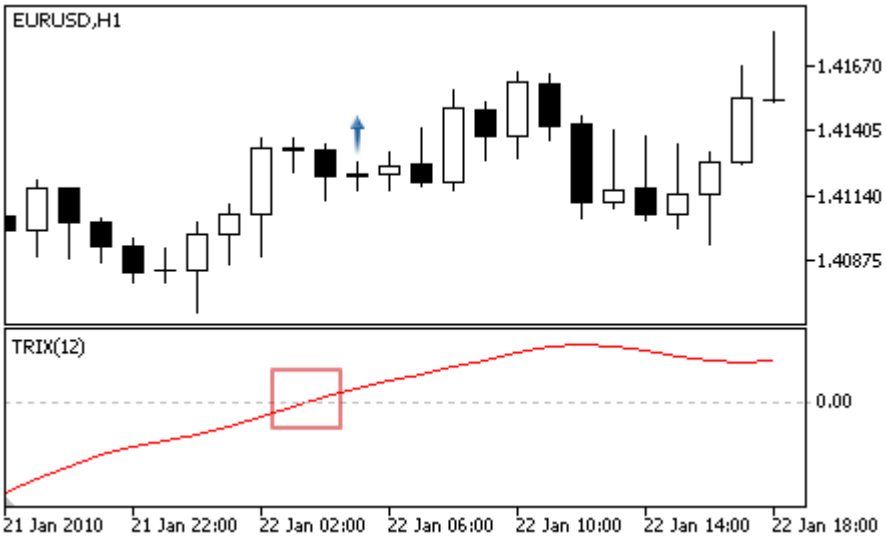
© 2000-2025, MetaQuotes Ltd.

Señales del oscilador Media Exponencial Triple

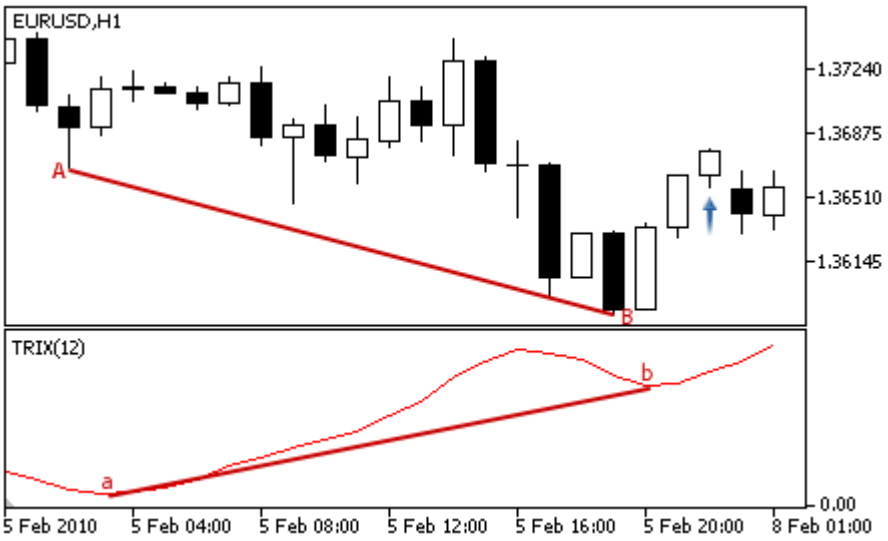
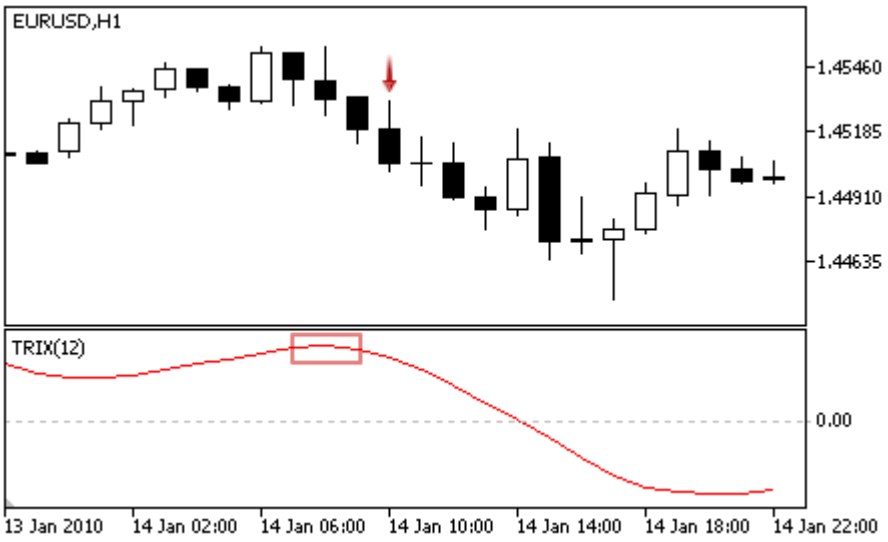
Este módulo de señales se basa en los modelos de mercado del oscilador Media Exponencial Triple. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

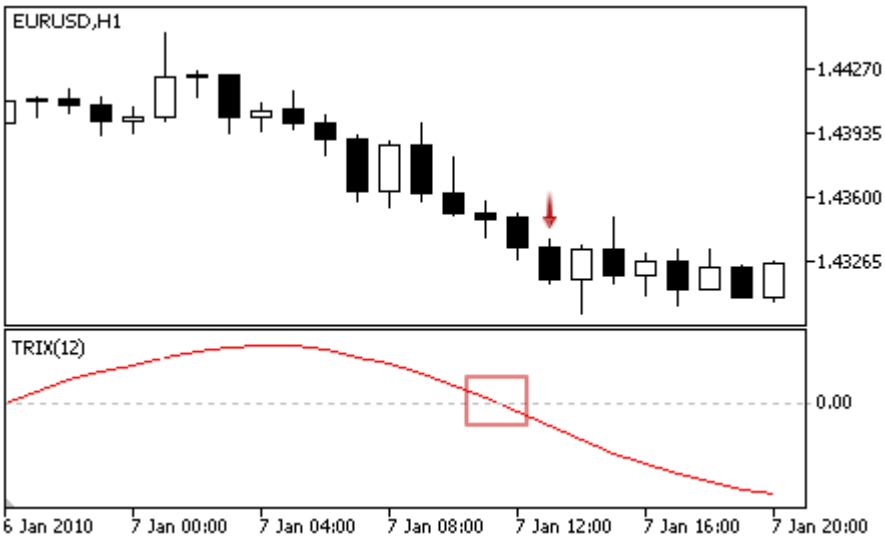
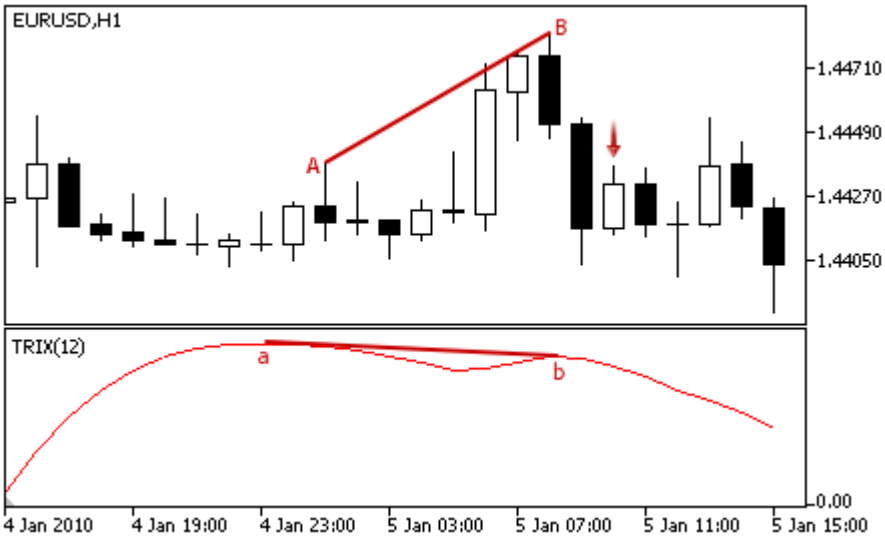
A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• **Inversión** — el oscilador se gira hacia arriba, subiendo en el barra analizada y cayendo en la anterior.  • **Cruce del nivel cero** — la línea principal está por encima del nivel cero en la barra analizada, y por debajo del nivel cero en la barra anterior. 

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	• **Divergencia** — el primer pico inferior analizado por el oscilador es mayor que el anterior, y el pico inferior del precio correspondiente es más bajo que el anterior. 
De venta	• **Inversión** — el oscilador se gira hacia abajo, cayendo en la barra analizada y subiendo en la barra anterior.  • **Cruce del nivel cero** — la línea principal está por debajo del nivel cero en la barra analizada, y por encima del nivel cero en la barra anterior.

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • **Divergencia** — el primer pico analizado del oscilador es menor que el anterior, y el pico del precio correspondiente es más alto que el pico anterior. 
No hay objeciones para comprar	El valor del oscilador crece en la barra analizada.
No hay objeciones para vender	El valor del oscilador cae en la barra analizada.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodTriX	Período de cálculo del oscilador.
Applied	La serie de precios utilizada en el cálculo del oscilador.

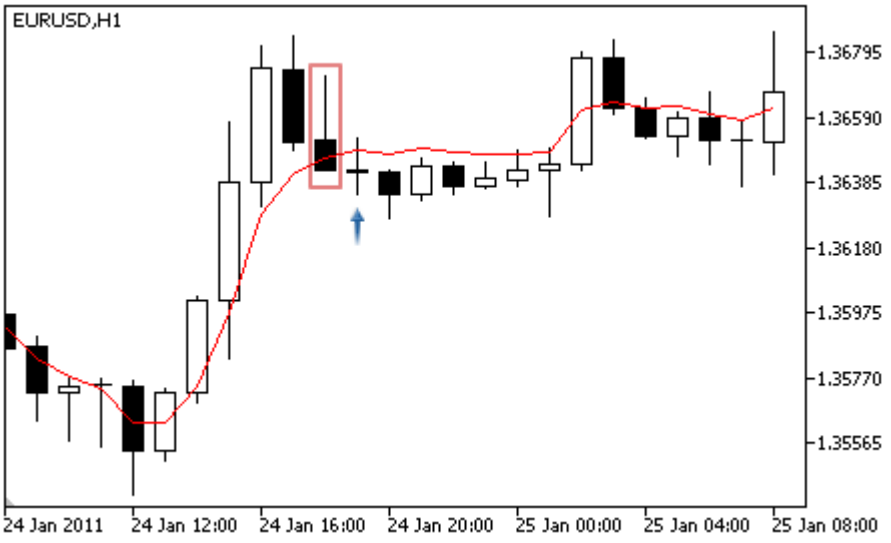
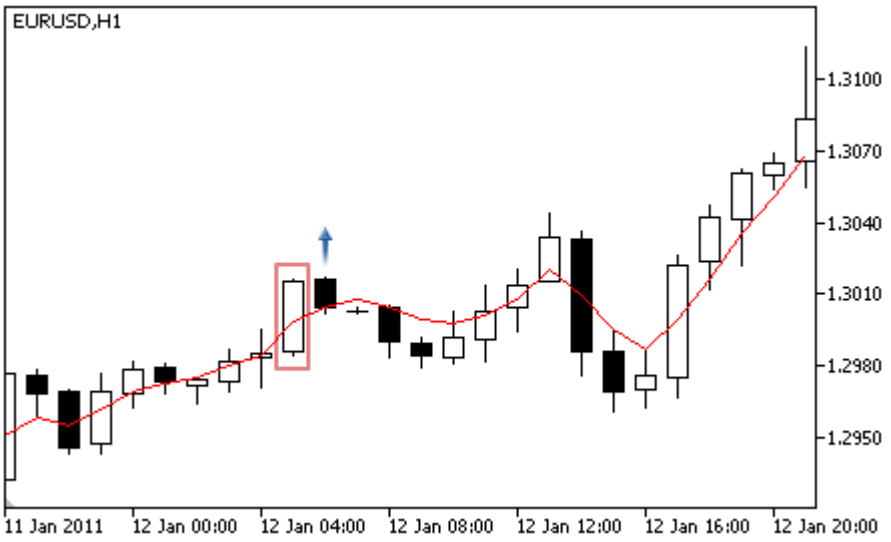
© 2000-2025, MetaQuotes Ltd.

Señales del indicador Media Móvil Exponencial Triple

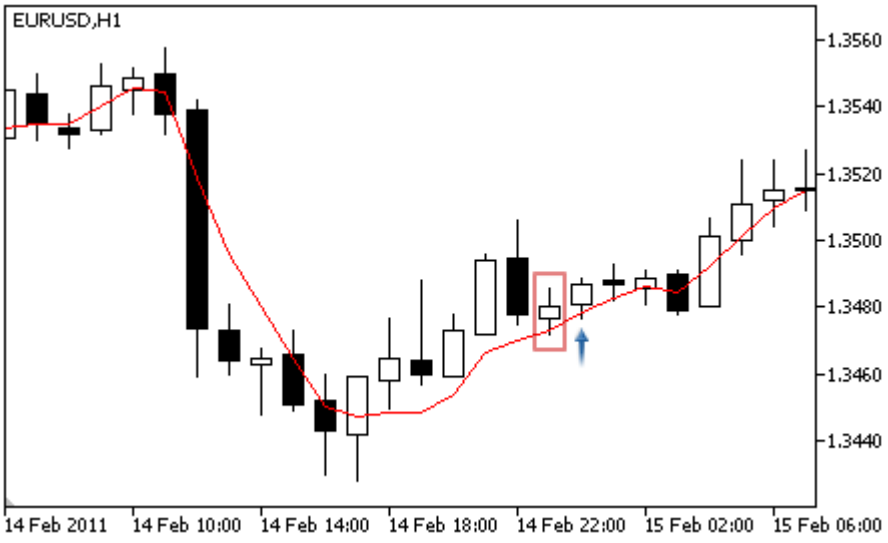
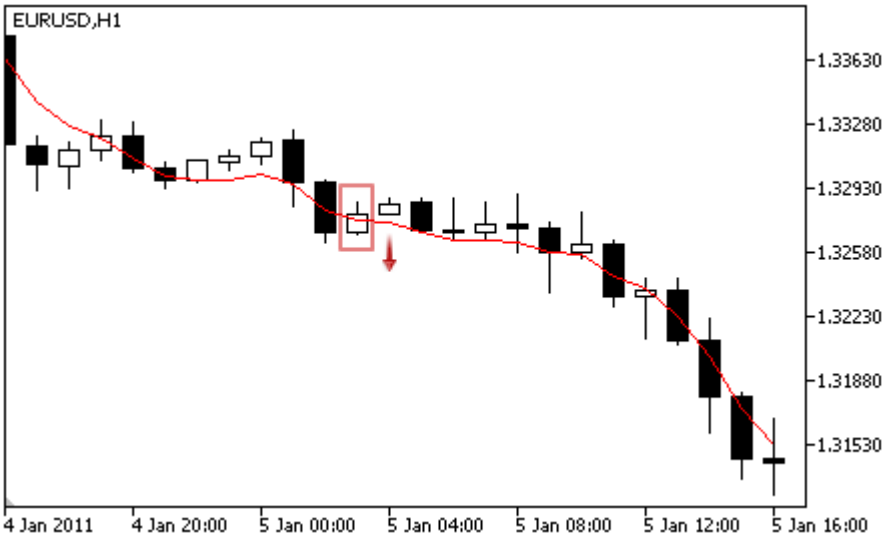
Este módulo de señales se basa en los modelos de mercado del indicador Media Móvil Exponencial Triple. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

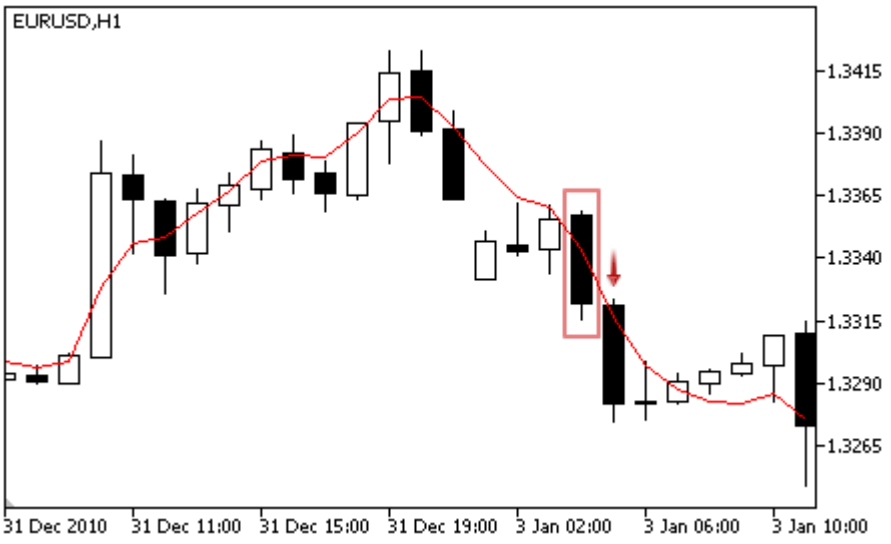
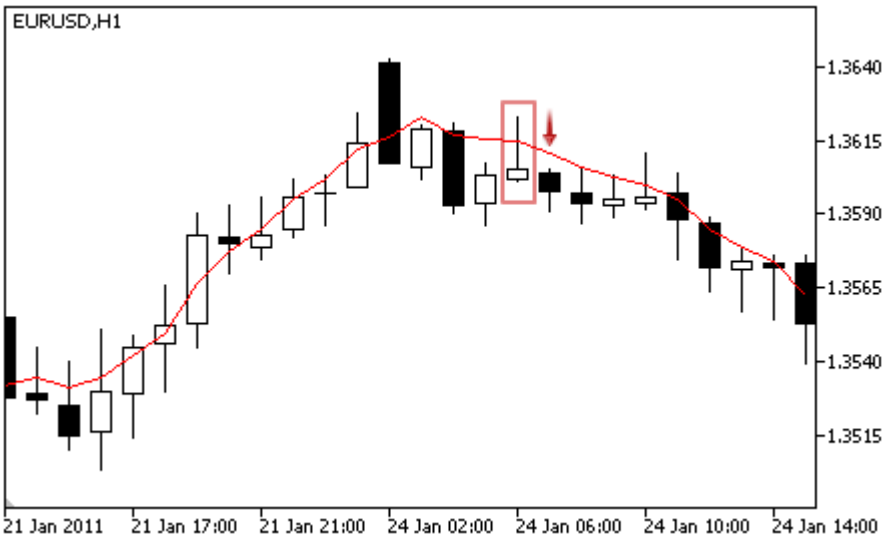
A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• El precio ha cruzado el indicador hacia abajo (el precio de apertura de la barra analizada está por encima del indicador, y el precio de cierre se encuentra por debajo del mismo) y el indicador sube (señal débil).  • **Cruce de la media móvil.** El precio ha cruzado el indicador hacia arriba (el precio de apertura de la barra analizada está por debajo del indicador, y el precio de cierre se encuentra por encima del mismo) y el indicador sube (señal fuerte). 

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	• La sombra inferior de la barra ha cruzado el indicador (los precios de apertura y cierre de la barra analizada están por encima del indicador, y el precio bajo se encuentra por debajo del mismo) y el indicador sube (señal débil). 
De venta	• El precio ha cruzado el indicador hacia arriba (el precio de apertura de la barra analizada está por debajo del indicador, y el precio de cierre se encuentra sobre el mismo) y el indicador cae (señal débil).  • **Cruce de la media móvil.** El precio ha cruzado el indicador hacia abajo (el precio de apertura de la barra analizada está por encima del indicador, y el precio de cierre se encuentra por debajo del mismo) y el indicador cae (señal fuerte).

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
	 • La sombra superior de la barra ha cruzado el indicador (los precios de apertura y cierre de la barra analizada están por debajo del indicador, y el precio alto se encuentra sobre el mismo) y el indicador cae (señal débil). 
No hay objeciones para comprar	El precio está por encima del indicador.
No hay objeciones para vender	El precio está por debajo del indicador.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

© 2000-2025, MetaQuotes Ltd.

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodMA	Período de promediación del indicador.
Shift	Desplazamiento del indicador a lo largo del eje temporal (en barras).
Method	Método de promediación.
Applied	La serie de precios utilizada para calcular el indicador.

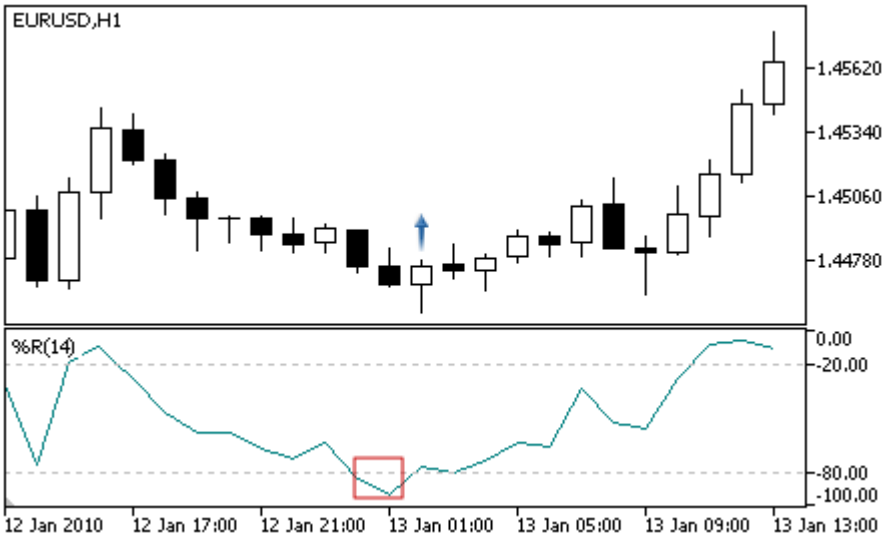
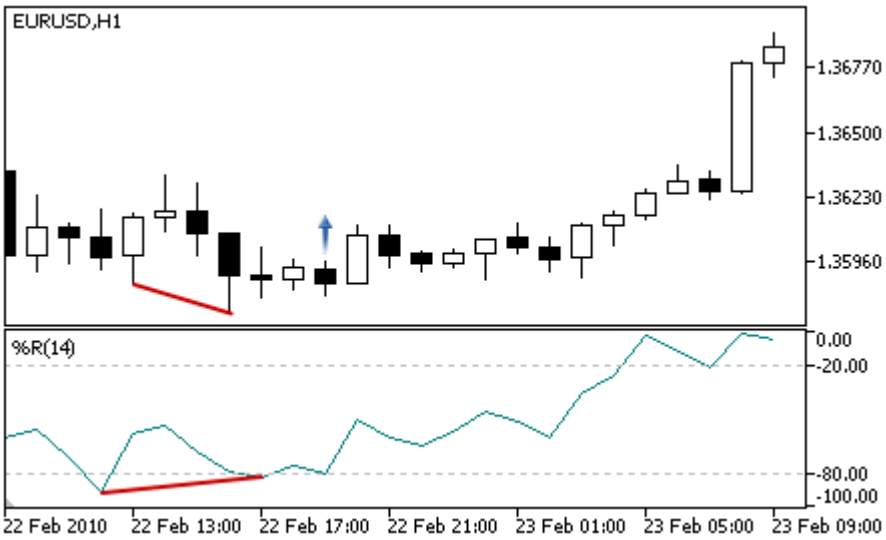
© 2000-2025, MetaQuotes Ltd.

Señales del oscilador Rango Porcentual de Williams

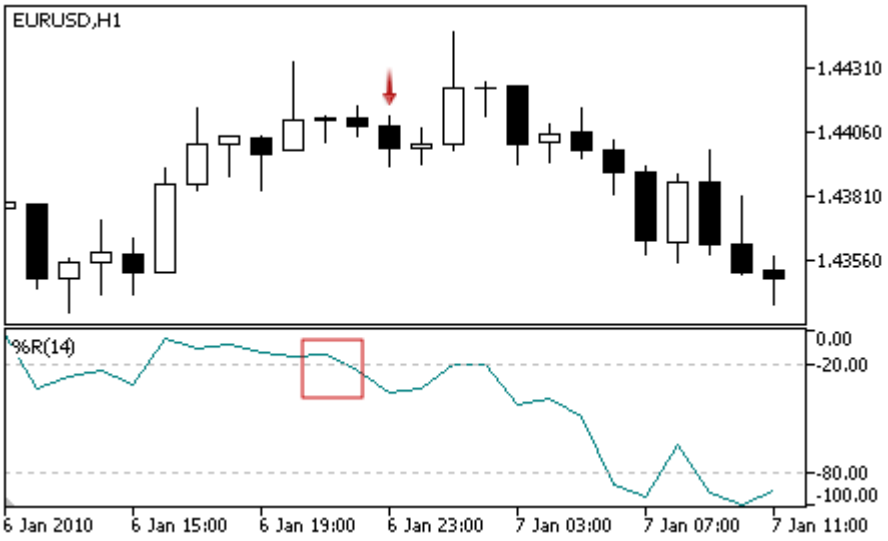
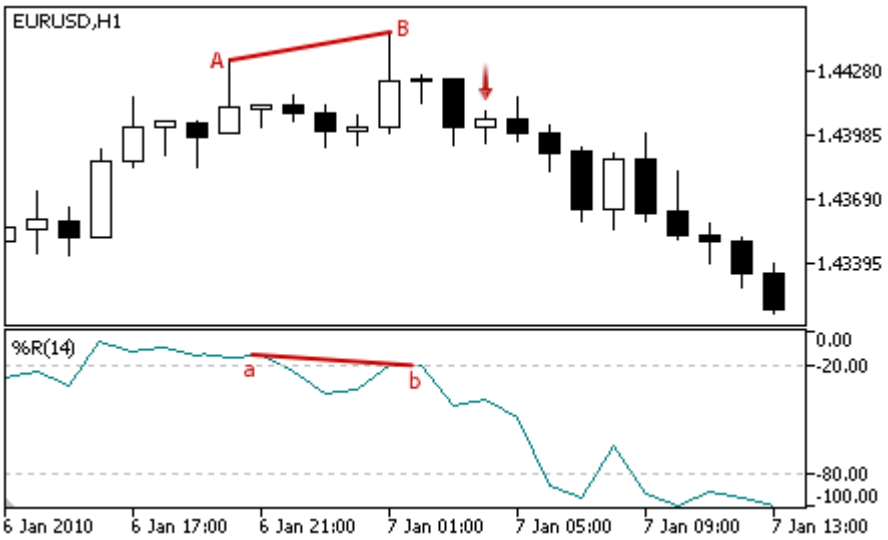
Este módulo de señales se basa en los modelos de mercado del oscilador Rango Porcentual de Williams. El mecanismo de toma de decisiones de trading basadas en las señales obtenidas en los módulos se describe en una sección aparte.

Condiciones para la generación de las señales

A continuación se describen las condiciones cuando el módulo pasa una señal a un Asesor Experto.

Tipo de señal	Descripción de las condiciones
De compra	• **Inversión por detrás del nivel de sobreventa** — el oscilador se gira hacia arriba, y su valor en la barra analizada está por detrás del nivel de sobreventa (el valor predeterminado es -20).  • **Divergencia** — el primer pico inferior analizado por el oscilador es mayor que el anterior, y el pico inferior del precio correspondiente es más bajo que el anterior. 

© 2000-2025, MetaQuotes Ltd.

Tipo de señal	Descripción de las condiciones
De venta	• **Inversión por detrás del nivel de sobrecompra** — el oscilador se gira hacia abajo, y su valor en la barra analizada está detrás del nivel de sobrecompra (el valor predeterminado es -80).  • **Divergencia** — el primer pico analizado del oscilador es menor que el anterior, y el pico del precio correspondiente es más alto que el pico anterior. 
No hay objeciones para comprar	El valor del oscilador crece en la barra analizada.
No hay objeciones para vender	El valor del oscilador cae en la barra analizada.

Nota

Dependiendo del modo de operación del Asesor Experto ("Cada tick" o "Sólo precios de apertura") la barra analizada es la actual (con índice 0), o bien la última barra que se haya formado (con índice 1).

© 2000-2025, MetaQuotes Ltd.

Recuerde que el oscilador Rango Porcentual de Williams tiene una escala invertida. Su valor máximo es -100, y el mínimo es 0.

Parámetros ajustables

Este módulo tiene estos parámetros ajustables:

Parámetro	Descripción
Weight	Peso de la señal del módulo en el intervalo de 0 a 1.
PeriodWPR	Período de cálculo del oscilador.

© 2000-2025, MetaQuotes Ltd.

Trailing Stop classes

Esta sección contiene detalles técnicos sobre las clases de trailing stop, y describe los componentes relevantes de la librería estándar MQL5.

El uso de estas clases le ahorrará tiempo al crear (y probar) sus estrategias de trading.

Las clases de las estrategias de trading de la librería estándar MQL5 se encuentran en el directorio Include\Expert\Trailing folder.

Clase	Descripción
CTrailingFixedPips	Esta clase implementa un algoritmo de Trailing Stop basado en puntos fijos
CTrailingMA	Esta clase implementa un algoritmo de Trailing Stop basado en los valores del indicador Media Móvil
CTrailingNone	Una clase stub que no utiliza ningún algoritmo de Trailing Stop
CTrailingPSAR	Esta clase implementa un algoritmo de Trailing Stop basado en los valores del indicador SAR Parabólico

© 2000-2025, MetaQuotes Ltd.

CTrailingFixedPips

La clase CTrailingFixedPips implementa un algoritmo de Trailing Stop basado en puntos fijos.

Si la posición tiene precio de Stop Loss, comprueba la distancia mínima de Stop Loss permitida al precio actual. Si su valor es inferior, ese nivel de Stop Loss sugiere establecer un nuevo precio de Stop Loss. En ese caso, si la posición tiene precio de Take Profit, se sugiere establecer un nuevo precio de Take Profit.

Si el Asesor Experto se inicializa con la bandera every_tick=false, entonces todas las operaciones se llevarán a cabo solamente en la nueva barra. En ese caso se puede utilizar el nivel de Take profit. Permite cerrar una posición abierta con el precio Take Profit antes de que la nueva barra se complete.

Descripción

La clase CTrailingFixedPips implementa un algoritmo de Trailing Stop basado en posiciones de trailing con puntos fijos.

Declaración

```
class CTrailingFixedPips: public CExpertTrailing
```

Título

```
   #include <Expert\Trailing\CTrailingFixedPips.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpertTrailing
      CTrailingFixedPips

Métodos de la clase

Inicialización	
StopLevel	Establece el valor del nivel Stop Loss
ProfitLevel	Establece el valor del nivel Take Profit
virtual ValidationSettings	Comprueba las configuraciones
Comprobación de los métodos de trailing	
virtual CheckTrailingStopLong	Comprueba las condiciones Trailing Stop de la posición larga
virtual CheckTrailingStopShort	Comprueba las condiciones Trailing Stop de la posición corta

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2025, MetaQuotes Ltd.

StopLevel

Establece el valor del nivel Stop Loss (en puntos).

```
void  StopLevel(
   int  stop_level      // Nivel Stop Loss
   )
```

Parámetros

stop_loss

[in] El valor del nivel Stop Loss (en puntos de 2/4-dígitos).

Nota

Si el nivel de Stop Loss es igual a 0, el Trailing Stop no se utiliza.

© 2000-2025, MetaQuotes Ltd.

ProfitLevel

Establece el valor del nivel de Take Profit (en puntos).

```
void  ProfitLevel(
   int  profit_level      // Nivel Take Profit
   )
```

Parámetros

profit_level

[in] El valor del nivel Take Profit (en puntos de 2/4-dígitos).

Nota

Si el nivel de Stop Loss es igual a 0, el Trailing Stop no se utiliza.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Comprueba las configuraciones.

```
virtual bool  ValidationSettings()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La función comprueba los niveles de Take Profit y Stop Loss. Los valores correctos son el 0 y los valores mayores que el stop mínimo para las órdenes stop del símbolo.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Comprueba las condiciones de Trailing Stop de la posición larga.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // Puntero al objeto CPositionInfo
   double&         sl,           // Precio Stop Loss
   double&         tp            // Precio Take Profit
   )
```

Parámetros

position

[in] Puntero al objeto CPositionInfo.

sl

[in][out] Variable de tipo precio Stop Loss.

tp

[in][out] Variable del precio Take Profit.

Valor devuelto

true si las condiciones se satisfacen; en caso contrario, false.

Nota

Si el nivel de Stop Loss es igual a 0, el Trailing Stop no se utiliza. Si la posición ya tiene precio de Stop Loss, se asume que su valor es el precio base; de lo contrario, se asume como precio base el precio de apertura de la posición.

Si el precio Bid actual es superior al precio base+nivel stop loss, se sugiere establecer un nuevo precio de Stop Loss. En este caso, si la posición ya tiene precio de Take Profit, se sugiere establecer el nuevo precio de Take Profit igual al precio Bid+nivel take proft.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Comprueba las condiciones Trailing Stop de la posición corta.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // Puntero al objeto CPositionInfo
   double&         sl,           // Precio Stop Loss
   double&         tp            // Precio Take Profit
   )
```

Parámetros

position

[in] Puntero al objeto CPositionInfo.

sl

[in][out] Variable de tipo precio Stop Loss.

tp

[in][out] Variable del precio Take Profit.

Valor devuelto

true si las condiciones se satisfacen; en caso contrario, false.

Nota

Si el nivel de Stop Loss es igual a 0, el Trailing Stop no se utiliza. Si la posición ya tiene precio de Stop Loss, se asume que su valor es el precio base; de lo contrario, se asume como precio base el precio de apertura de la posición.

Si el precio Ask actual es inferior al precio base-nivel stop loss, se sugiere establecer un nuevo precio Stop Loss. En este caso, si la posición ya tiene un precio Take Profit, se sugiere establecer un nuevo precio de Take Profit igual al precio Ask-nivel take profit.

© 2000-2025, MetaQuotes Ltd.

CTrailingMA

La clase CTrailingMA implementa un algoritmo de Trailing Stop basado en los valores del indicador media móvil.

Descripción

La clase CTrailingMA implementa un algoritmo de Trailing Stop basado en los valores del indicador media móvil de la barra anterior (completa).

Declaración

```
class CTrailingMA: public CExpertTrailing
```

Título

```
   #include <Expert\Trailing\TrailingMA.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpertTrailing
      CTrailingMA

Métodos de la clase

Inicialización	
Period	Establece el periodo de la media móvil
Shift	Establece el desplazamiento de la media móvil
Method	Establece el método de suavizado de la media móvil
Applied	Establece el precio aplicado de la media móvil
virtual InitIndicators	Inicializa los indicadores y las series temporales
virtual ValidationSettings	Comprueba las configuraciones
Comprobación de los métodos de trailing	
virtual CheckTrailingStopLong	Comprueba las condiciones Trailing Stop de la posición larga
virtual CheckTrailingStopShort	Comprueba las condiciones Trailing Stop de la posición corta

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

Period

Establece el periodo de la media móvil.

```
void  Period(
   int  period      // Periodo de suavizado
   )
```

Parámetros

period

[in] Periodo de la media móvil.

© 2000-2025, MetaQuotes Ltd.

Shift

Establece el desplazamiento de la media móvil.

```
void  Shift(
   int  shift      // Desplazamiento
   )
```

Parámetros

shift

[in] Desplazamiento de la media móvil.

© 2000-2025, MetaQuotes Ltd.

Method

Establece el método de suavizado de la media móvil.

```
void  Method(
   ENUM_MA_METHOD  method      // Método de suavizado
   )
```

Parámetros

method

[in] Método de suavizado del indicador de media móvil.

© 2000-2025, MetaQuotes Ltd.

Applied

Establece el precio aplicado de la media móvil.

```
void  Applied(
   ENUM_APPLIED_PRICE  applied      // Precio aplicado
   )
```

Parámetros

applied

[in] Precio aplicado de la media móvil.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

Inicializa los indicadores y las series temporales.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // Puntero a la colección CIndicators
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales (miembro de la clase CExpert).

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Comprueba las configuraciones.

```
virtual bool  ValidationSettings()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La función comprueba el período de la media móvil, los valores correctos son positivos.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Comprueba las condiciones de Trailing Stop de la posición larga.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // Puntero al objeto CPositionInfo
   double&         sl,           // Precio Stop Loss
   double&         tp            // Precio Take Profit
   )
```

Parámetros

position

[in] Puntero al objeto CPositionInfo.

sl

[in][out] Variable de tipo precio Stop Loss.

tp

[in][out] Variable del precio Take Profit.

Valor devuelto

true si las condiciones se satisfacen; en caso contrario, false.

Nota

Calcula el precio máximo permitido de Stop Loss más cercano al precio actual, calculando el precio de Stop Loss con los valores del indicador de media móvil de la barra (completa) anterior.

Si la posición ya tiene precio de Stop Loss, se asume como precio base ese valor; de lo contrario, el precio base es el precio de apertura de la posición.

Si el precio de Stop Loss calculado es superior al precio base, e inferior al precio máximo permitido de Stop Loss, se sugiere establecer el nuevo precio de Stop Loss.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Comprueba las condiciones Trailing Stop de la posición corta.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // Puntero al objeto CPositionInfo
   double&         sl,           // Precio Stop Loss
   double&         tp            // Precio Take Profit
   )
```

Parámetros

position

[in] Puntero al objeto CPositionInfo.

sl

[in][out] Variable de tipo precio Stop Loss.

tp

[in][out] Variable del precio Take Profit.

Valor devuelto

true si las condiciones se satisfacen; en caso contrario, false.

Nota

Calcula el precio mínimo permitido de Stop Loss más cercano al precio actual, calculando el precio de Stop Loss con los valores del indicador de media móvil de la barra anterior (completa).

Si la posición ya tiene precio de Stop Loss, se asume como precio base ese valor; de lo contrario, el precio base es el precio de apertura de la posición.

Si el precio de Stop Loss calculado es superior al precio base, e inferior al precio mínimo permitido de Stop Loss, se sugiere establecer el nuevo precio de Stop Loss.

© 2000-2025, MetaQuotes Ltd.

CTrailingNone

CTrailingNone es una clasestub. Esta clase tiene que utilizarse en la inicialización del objeto Trailng si su estrategia no utiliza Trailing Stop.

Descripción

La clase CTrailingNone no implementa ningún algoritmo de Trailing Stop. Los métodos de comprobación de las condiciones de Trailing Stop siempre devuelven false.

Declaración

```
class CTrailingNone: public CExpertTrailing
```

Título

```
   #include <Expert\Trailing\TrailingNone.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpertTrailing
      CTrailingNone

Métodos de la clase

Comprobación de los métodos de trailing	
virtual CheckTrailingStopLong	Método stub que comprueba las condiciones de Trailing Stop de la posición larga
virtual CheckTrailingStopShort	Método stub que comprueba las condiciones de Trailing Stop de la posición corta

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos heredados de la clase CExpertTrailing

CheckTrailingStopLong, CheckTrailingStopShort

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Comprueba las condiciones de Trailing Stop de la posición larga.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // Puntero al objeto CPositionInfo
   double&         sl,           // Precio Stop Loss
   double&         tp            // Precio Take Profit
   )
```

Parámetros

position

[in] Puntero al objeto CPositionInfo.

sl

[in][out] Variable de tipo precio Stop Loss.

tp

[in][out] Variable del precio Take Profit.

Valor devuelto

true si las condiciones se satisfacen; en caso contrario, false.

Nota

La función siempre devuelve false.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Comprueba las condiciones Trailing Stop de la posición corta.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // Puntero al objeto CPositionInfo
   double&         sl,           // Precio Stop Loss
   double&         tp            // Precio Take Profit
   )
```

Parámetros

position

[in] Puntero al objeto CPositionInfo.

sl

[in][out] Variable de tipo precio Stop Loss.

tp

[in][out] Variable del precio Take Profit.

Valor devuelto

true si las condiciones se satisfacen; en caso contrario, false.

Nota

La función siempre devuelve false.

© 2000-2025, MetaQuotes Ltd.

CTrailingPSAR

La clase CTrailingPSAR implementa un algoritmo de Trailing Stop basado en los valores del indicador SAR Parabólico.

Descripción

La clase CTrailingPSAR implementa un algoritmo de Trailing Stop basado en los valores del indicador SAR Parabólico de la barra anterior (completada).

Declaración

```
class CTrailingPSAR: public CExpertTrailing
```

Título

```
   #include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpertTrailing
      CTrailingPSAR

Métodos de la clase

Inicialización	
Step	Establece el valor del paso del indicador SAR Parabólico
Maximum	Establece el valor máximo del indicador SAR Parabólico
virtual InitIndicators	Inicializa los indicadores y las series temporales
Comprobación de los métodos de trailing	
virtual CheckTrailingStopLong	Comprueba las condiciones de trailing stop de la posición larga
virtual CheckTrailingStopShort	Comprueba las condiciones de trailing stop de la posición corta

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Step

Establece el valor de paso del indicador SAR Parabólico.

```
void  Step(
   double  step      // Paso
   )
```

Parámetros

step

[in] El valor de paso del indicador SAR Parabólico.

© 2000-2025, MetaQuotes Ltd.

Maximum

Establece el valor máximo del indicador SAR Parabólico.

```
void  Maximum(
   double  maximum      // Máximo
   )
```

Parámetros

maximum

[in] El valor máximo del indicador SAR Parabólico.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

Inicializa los indicadores y las series temporales.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // Puntero a la colección CIndicators
   )
```

Parámetros

indicators

[in] Puntero a la colección de indicadores y series temporales (miembro de la clase CExpert).

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Comprueba las condiciones de Trailing Stop de la posición larga.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // Puntero
   double&         sl,           // Enlace
   double&         tp            // Enlace
   )
```

Parámetros

position

[in] Puntero al objeto CPositionInfo.

sl

[in][out] Variable de tipo precio Stop Loss.

tp

[in][out] Variable del precio Take Profit.

Valor devuelto

true si las condiciones se satisfacen; en caso contrario, false.

Nota

Calcula el precio máximo permitido de Stop Loss, más cercano al precio actual, calculando el precio de Stop Loss con los valores del indicador SAR Parabólico de la barra anterior (completa).

Si la posición ya tiene precio de Stop Loss, se asume que su valor es el precio base; de lo contrario, se asume como precio base el precio de apertura de la posición.

Si el precio de Stop Loss calculado es superior al precio base, e inferior al precio máximo permitido de Stop Loss, se sugiere establecer el nuevo precio de Stop Loss.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Comprueba las condiciones Trailing Stop de la posición corta.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // Puntero
   double&         sl,           // Enlace
   double&         tp            // Enlace
   )
```

Parámetros

position

[in] Puntero al objeto CPositionInfo.

sl

[in][out] Variable de tipo precio Stop Loss.

tp

[in][out] Variable del precio Take Profit.

Valor devuelto

true si las condiciones se satisfacen; en caso contrario, false.

Nota

Calcula el precio mínimo permitido de Stop Loss, más cercano al precio actual, calculando el precio de Stop Loss con los valores del indicador SAR Parabólico de la barra anterior (completa).

Si la posición ya tiene precio de Stop Loss, se asume que su valor es el precio base; de lo contrario, se asume como precio base el precio de apertura de la posición.

Si el precio de Stop Loss calculado es superior al precio base, e inferior al precio mínimo permitido de Stop Loss, se sugiere establecer el nuevo precio de Stop Loss.

© 2000-2025, MetaQuotes Ltd.

Clases de gestión del dinero

Esta sección contiene detalles técnicos sobre las clases de gestión del dinero y del riesgo, y describe los componentes relevantes de la librería estándar MQL5.

El uso de estas clases le ahorrará tiempo al crear (y probar) sus estrategias de trading.

Las clases de gestión del dinero y del riesgo de la librería estándar MQL5 se encuentran en el directorio Include\Expert\Money\.

Clase	Descripción
CMoneyFixedLot	Esta clase implementa un algoritmo de gestión del dinero basado en el tamaño predefinido de lote fijo.
CMoneyFixedMargin	Esta clase implementa un algoritmo de gestión del dinero basado en el margen fijo predefinido.
CMoneyFixedRisk	Esta clase implementa un algoritmo de gestión del dinero basado en el riesgo predefinido.
CMoneyNone	Esta clase implementa un algoritmo de gestión del dinero basado en el tamaño de lote mínimo permitido.
CMoneySizeOptimized	Esta clase implementa un algoritmo de gestión del dinero basado en el el tamaño de lote variable que depende de los resultados de las operaciones anteriores.

© 2000-2025, MetaQuotes Ltd.

CMoneyFixedLot

La clase CMoneyFixedLot implementa un algoritmo de trading basado en el tamaño predefinido de lote fijo.

Descripción

CMoneyFixedLot implementa un algoritmo de trading basado en el tamaño predefinido de lote fijo.

Declaración

```
class CMoneyFixedLot: public CExpertMoney
```

Título

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedLot

Métodos de la clase

Inicialización	
Lots	Establece el volumen de trading
virtual ValidationSettings	Comprueba las configuraciones
Métodos de gestión del dinero y del riesgo	
virtual CheckOpenLong	Obtiene el volumen de trading de la posición larga
virtual CheckOpenShort	Obtiene el volumen de trading de la posición corta

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos heredados de la clase CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

Lots

Establece el volumen de trading (en lotes).

```
void  Lots(
   double  lots      // Lotes
   )
```

Parámetros

lots

[in] Volumen de trading (en lotes).

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Comprueba las configuraciones.

```
virtual bool  ValidationSettings()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

Comprueba el volumen especificado para su corrección.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Obtiene el volumen de trading de la posición larga.

```
virtual double  CheckOpenLong(
   double  price,     // Precio
   double  sl         // Precio Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición larga.

Nota

La función siempre devuelve el volumen fijo definido en el método Lots.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Obtiene el volumen de la posición corta.

```
virtual double  CheckOpenShort(
   double  price,     // Precio
   double  sl         // Precio Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición corta.

Nota

La función siempre devuelve el volumen fijo definido en el método Lots.

© 2000-2025, MetaQuotes Ltd.

CMoneyFixedMargin

CMoneyFixedMargin implementa un algoritmo de gestión del dinero basado en el margen fijo predefinido.

Descripción

CMoneyFixedMargin implementa un algoritmo de gestión del dinero basado en el margen fijo predefinido.

Declaración

```
class CMoneyFixedMargin: public CExpertMoney
```

Título

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedMargin

Métodos de la clase

Métodos de gestión del dinero y del riesgo	
virtual CheckOpenLong	Obtiene el volumen de trading de la posición larga
virtual CheckOpenShort	Obtiene el volumen de trading de la posición corta

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos heredados de la clase CExpertMoney

Percent, ValidationSettings, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Obtiene el volumen de trading de la posición larga.

```
virtual double  CheckOpenLong(
   double  price,     // Precio
   double  sl         // Precio Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición larga.

Nota

La función devuelve el volumen de la posición larga, utilizando el margen fijo. El margen se define en el parámetro Percent de la clase base CExpertMoney.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Obtiene el volumen de la posición corta.

```
virtual double  CheckOpenShort(
   double  price,     // Precio
   double  sl         // Precio Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición corta.

Nota

La función devuelve el volumen de la posición corta, utilizando el margen fijo. El margen se define en el parámetro Percent de la clase base CExpertMoney.

© 2000-2025, MetaQuotes Ltd.

CMoneyFixedRisk

CMoneyFixedRisk implementa un algoritmo de gestión del dinero basado en el riesgo predefinido.

Descripción

CMoneyFixedRisk implementa un algoritmo de gestión del dinero basado en el riesgo predefinido.

Declaración

```
class CMoneyFixedRisk: public CExpertMoney
```

Título

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedRisk

Métodos de la clase

Métodos de gestión del dinero y del riesgo	
virtual CheckOpenLong	Obtiene el volumen de trading de la posición larga
virtual CheckOpenShort	Obtiene el volumen de trading de la posición corta

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos heredados de la clase CExpertMoney

Percent, ValidationSettings, CheckReverse

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Obtiene el volumen de trading de la posición larga.

```
virtual double  CheckOpenLong(
   double  price,     // Precio
   double  sl         // Precio Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición larga.

Nota

La función devuelve el volumen de la posición larga, utilizando el riesgo fijo. El riesgo se define en el parámetro Percent de la clase base CExpertMoney.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Obtiene el volumen de la posición corta.

```
virtual double  CheckOpenShort(
   double  price,     // Precio
   double  sl         // Precio Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición corta.

Nota

La función devuelve el volumen de la posición corta, utilizando el riesgo fijo. El riesgo se define en el parámetro Percent de la clase base CExpertMoney.

© 2000-2025, MetaQuotes Ltd.

CMoneyNone

CMoneyNone implementa un algoritmo de trading basado en el lote mínimo permitido.

Descripción

CMoneyNone implementa un algoritmo de trading basado en el lote mínimo permitido.

Declaración

```
class CMoneyNone: public CExpertMoney
```

Título

```
#include <Expert\Money\MoneyNone.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpertMoney
      CMoneyNone

Métodos de la clase

Inicialización	
virtual ValidationSettings	Comprueba las configuraciones
Métodos de gestión del dinero y del riesgo	
virtual CheckOpenLong	Obtiene el volumen de trading de la posición larga
virtual CheckOpenShort	Obtiene el volumen de trading de la posición corta

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos heredados de la clase CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Comprueba las configuraciones.

```
virtual bool  ValidationSettings()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

La función devuelve siempre true.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Obtiene el volumen de trading de la posición larga.

```
virtual double  CheckOpenLong(
   double  price,     // Precio
   double  sl         // Precio Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición larga.

Nota

La función devuelve siempre el tamaño de lote mínimo.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Obtiene el volumen de trading de la posición larga.

```
virtual double  CheckOpenShort(
   double  price,     // Precio
   double  sl         // Precio Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición corta.

Nota

La función devuelve siempre el tamaño de lote mínimo.

© 2000-2025, MetaQuotes Ltd.

CMoneySizeOptimized

CMoneySizeOptimized implementa un algoritmo de trading basado en el el tamaño de lote variable que depende de los resultados de las operaciones anteriores.

Descripción

CMoneySizeOptimized implementa un algoritmo de trading basado en el el tamaño de lote variable que depende de los resultados de las operaciones anteriores.

Declaración

```
class CMoneySizeOptimized: public CExpertMoney
```

Título

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

Jerarquía de herencia

CObject
  CExpertBase
    CExpertMoney
      CMoneySizeOptimized

Métodos de la clase

Inicialización	
DecreaseFactor	Establece el valor del factor de reducción.
virtual ValidationSettings	Comprueba las configuraciones
Métodos de gestión del dinero y del riesgo	
virtual CheckOpenLong	Obtiene el volumen de trading de la posición larga
virtual CheckOpenShort	Obtiene el volumen de trading de la posición corta

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos heredados de la clase CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

DecreaseFactor

Establece el valor del factor de reducción.

```
void  DecreaseFactor(
   double  decrease_factor      // Factor de reducción
   )
```

Parámetros

decrease_factor

[in] Factor de reducción.

Nota

El factor de reducción define el coeficiente de decrecimiento del volumen (comparado con el volumen de la posición anterior) en caso de que se produzcan operaciones perdedoras consecutivas.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Comprueba las configuraciones.

```
virtual bool  ValidationSettings()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

Si el factor de disminución es negativo devuelve false; en caso contrario, devuelve true.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Obtiene el volumen de trading de la posición larga.

```
virtual double  CheckOpenLong(
   double  price,     // Precio
   double  sl         // Precio Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición larga.

Nota

La función devuelve el volumen de la posición larga, que depende de los resultados de las operaciones anteriores.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Obtiene el volumen de la posición corta.

```
virtual double  CheckOpenShort(
   double  price,     // Precio
   double  sl         // Precio Stop Loss
   )
```

Parámetros

price

[in] Precio.

sl

[in] Precio Stop Loss.

Valor devuelto

Volumen de la posición larga.

Nota

La función devuelve el volumen de la posición corta, que depende de los resultados de las operaciones anteriores.

© 2000-2025, MetaQuotes Ltd.

Clases de creación de los Paneles de Control y de los Cuadros de Diálogo

Esta sección contiene detalles técnicos del funcionamiento de las clases de creación de paneles de control, así como descripciones de componentes relevantes de la Librería Estándar del lenguaje MQL5.

Estas clases le ayudarán a ahorrar tiempo en el momento de crear los paneles de control de sus programas MQL5 (Asesores Expertos e indicadores).

Los controles de la Librería Estándar MQL5 se encuentran en la carpeta de datos del terminal cliente, en MQL5\Include\Controls.

Podrá encontrar ejemplos de trabajo con las clases en los artículos:

- Cómo crear un panel gráfico de cualquier nivel de complejidad
- Mejorando el trabajo con paneles: cómo añadir transparencia, cambiar el color del fondo y heredar a partir de CAppDialog/CWndClient
- Cómo añadir rápidamente un panel de control a un indicador o asesor
- Cree sus propios paneles gráficos en MQL5
- Crear Paneles de Control Activos en MQL5 para Trading

El Asesor Experto de ejemplo ilustra el funcionamiento de estas clases, y se encuentra en la carpeta MQL5\Expert\Examples\Controls.

Estructuras auxiliares	Descripción
CRect	Estructura del área rectangular
CDateTime	Estructura para trabajar con fechas y horas

Clases base	Descripción
CWnd	Clase base de los controles
CWndObj	Clase base de los controles y de los cuadros de diálogo
CWndContainer	Clase base de los controles complejos (que contienen controles dependientes)

Controles sencillos	Descripción
CLabel	Control basado en el objeto gráfico "Text label" (Etiqueta de texto)
CBmpButton	Control basado en el objeto gráfico "Bitmap label" (Etiqueta bitmap)
CButton	Control basado en el objeto gráfico "Button" (Botón)

© 2000-2025, MetaQuotes Ltd.

Controles sencillos	Descripción
CEdit	Control basado en el objeto gráfico "Edit field" (Campo de edición)
CPanel	Control basado en "Rectangle label" (Etiqueta rectangular)
CPicture	Control basado en "Bitmap label" (Etiqueta bitmap)

Complex controls	Descripción
CScroll	Clase base de la barra de desplazamiento
CScrollV	Barra de desplazamiento vertical
CScrollH	Barra de desplazamiento horizontal
CWndClient	Clase base de la zona cliente con barras de desplazamiento
CListView	ListView
CComboBox	ComboBox
CCheckBox	CheckBox
CCheckGroup	CheckGroup
CRadioButton	RadioButton
CRadioGroup	RadioGroup
CSpinEdit	SpinEdit
CDialog	Dialog
CAppDialog	Cuadro de diálogo de la aplicación

© 2000-2025, MetaQuotes Ltd.

CRect

CRect es una clase del área rectangular del gráfico.

Descripción

CRect es una clase del área, definida por las coordenadas de las esquinas superior izquierda e inferior derecha de un rectángulo de coordenadas cartesianas.

Declaración

```
class CRect
```

Título

```
#include <Controls\Rect.mqh>
```

Métodos de la clase

Propiedades	
Left	Obtiene/Establece la coordenada X de la esquina superior izquierda
Top	Obtiene/Establece la coordenada Y de la esquina superior izquierda
Right	Obtiene/Establece la coordenada X de la esquina inferior derecha
Bottom	Obtiene/Establece la coordenada Y de la esquina inferior derecha
Width	Obtiene/Establece la anchura
Height	Obtiene/Establece la altura
SetBound	Establece las nuevas coordenadas de la clase CRect
Move	Establece las nuevas coordenadas de la clase CRect
Shift	Realiza el desplazamiento relativo de las coordenadas CRect
Contains	Comprueba si el punto está dentro del área de la clase CRect
Métodos adicionales	
Format	Obtiene las coordenadas del área en formato string

© 2000-2025, MetaQuotes Ltd.

Left (Método Get)

Obtiene la coordenada X de la esquina superior izquierda.

```
int  Left()
```

Valor devuelto

Coordenada X de la esquina superior izquierda.

Left (Método Set)

Establece la coordenada X de la esquina superior izquierda.

```
void  Left(
   const int  x      // nueva coordenada x
   )
```

Parámetros

x

[in] Nueva coordenada X de la esquina superior izquierda.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Top (Método Get)

Obtiene la coordenada Y de la esquina superior izquierda.

```
int  Top()
```

Valor devuelto

La coordenada Y de la esquina superior izquierda.

Top (Método Set)

Establece la coordenada Y de la esquina superior izquierda.

```
void  Top(
   const int  y      // coordenada y
   )
```

Parámetros

y

[in] Nueva coordenada Y de la esquina superior izquierda.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Right (Método Get)

Obtiene la coordenada X de la esquina inferior derecha.

```
int  Right()
```

Valor devuelto

Coordenada X de la esquina inferior derecha.

Right (Método Set)

Establece la coordenada Y de la esquina inferior derecha.

```
void  Right(
   const int  x      // coordenada x
   )
```

Parámetros

x

[in] Nueva coordenada X de la esquina inferior derecha.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Bottom (Método Get)

Obtiene la coordenada Y de la esquina inferior derecha.

```
int  Bottom()
```

Valor devuelto

Coordenada Y de la esquina inferior derecha.

Bottom (Método Set)

Establece la coordenada Y de la esquina inferior derecha.

```
void  Bottom(
   const int  y      // coordenada y
   )
```

Parámetros

y

[in] Nueva coordenada Y de la esquina inferior derecha.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Width (Método Get)

Obtiene la anchura del área.

```
int  Width()
```

Valor devuelto

Anchura del área.

Width (Método Set)

Establece la nueva anchura del área.

```
virtual bool  Width(
   const int  w      // anchura
   )
```

Parámetros

w

[in] Nueva anchura.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Height (Método Get)

Obtiene la altura del área.

```
int  Height()
```

Valor devuelto

Altura del área.

Height (Método Set)

Establece la nueva altura del área.

```
virtual bool  Height(
   const int  h      // altura
   )
```

Parámetros

h

[in] Nueva altura.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

SetBound

Establece las nuevas coordenadas del área utilizando las coordenadas de la clase CRect.

```
void  SetBound(
   const & CRect  rect      // Clase CRect
   )
```

Valor devuelto

Ninguno.

SetBound

Establece las nuevas coordenadas del área.

```
void  SetBound(
   const int  l      // izquierda
   const int  t      // superior
   const int  r      // derecha
   const int  b      // abajo
   )
```

Parámetros

l

[in] Coordenada X de la esquina superior izquierda.

t

[in] Coordenada Y de la esquina superior izquierda.

r

[in] Coordenada X de la esquina inferior derecha.

b

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Move

Establece las nuevas coordenadas de la clase CRect.

```
void  Move(
   const int  x,      // coordenada X
   const int  y       // coordenada Y
   )
```

Parámetros

x

[in] Nueva coordenada X.

y

[in] Nueva coordenada Y.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Shift

Realiza el desplazamiento relativo de las coordenadas de la clase CRect.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Parámetros

dx

[in] Delta X.

dy

[in] Delta Y.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Contains

Comprueba si el punto está dentro del área de la clase CRect.

```
bool  Contains(
   const int  x,      // coordenada X
   const int  y       // coordenada Y
   )
```

Parámetros

x

[in] coordenada X.

y

[in] coordenada Y.

Valor devuelto

true, si el punto está dentro del área (incluyendo los bordes), false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Format

Obtiene las coordenadas del área en formato string.

```
string  Format(
   string &  fmt,     // formato
   ) const
```

Parámetros

fmt

[in] Cadena con formato.

Valor devuelto

Cadena con las coordenadas del área.

© 2000-2025, MetaQuotes Ltd.

CDateTime

CDateTime es una estructura para trabajar con fechas y horas.

Descripción

CDateTime es una estructura derivada de MqlDateTime, utilizada para realizar operaciones con fechas y horas en los controles.

Declaración

```
struct CDateTime
```

Título

```
#include <Tools\DateTime.mqh>
```

Métodos de la clase

Propiedades	
MonthName	Obtiene el nombre del mes
ShortMonthName	Obtiene el nombre corto del mes
DayName	Obtiene el nombre del día de la semana
ShortDayName	Obtiene el nombre corto del día de la semana
DaysInMonth	Obtiene el número de días de un mes
Métodos Get/Set	
DateTime	Obtiene/Establece la fecha y la hora
Date	Establece la fecha
Time	Establece la hora
Sec	Establece los segundos
Min	Establece los minutos
Hour	Establece la hora
Day	Establece el día del mes
Mon	Establece el mes
Year	Establece el año
Métodos adicionales	
SecDec	Resta el número de segundos especificado
SecInc	Añade el número de segundos especificado
MinDec	Resta el número de minutos especificado

© 2000-2025, MetaQuotes Ltd.

Propiedades	
MinInc	Añade el número de minutos especificado
HourDec	Resta el número de horas especificado
HourInc	Añade el número de horas especificado
DayDec	Resta el número de días especificado
DayInc	Añade el número de días especificado
MonDec	Resta el número de meses especificado
MonInc	Añade el número de meses especificado
YearDec	Resta el número de años especificado
YearInc	Añade el número de años especificado

© 2000-2025, MetaQuotes Ltd.

MonthName

Obtiene el nombre del mes.

```
string  MonthName()  const
```

Obtiene el nombre del mes por medio de un índice.

```
string  MonthName(
   const int       num          // índice del mes
   )  const
```

Parámetros

num

[in] Índice del mes (1-12).

Valor devuelto

Nombre del mes.

© 2000-2025, MetaQuotes Ltd.

ShortMonthName

Obtiene el nombre corto del mes.

```
string  ShortMonthName()  const
```

Obtiene el nombre corto del mes por medio de un índice.

```
string  ShortMonthName(
   const int      num         // índice del mes
   )  const
```

Parámetros

num

[in] Índice del mes (1-12).

Valor devuelto

Nombre corto del mes.

© 2000-2025, MetaQuotes Ltd.

DayName

Obtiene el nombre del día de la semana.

```
string  DayName()  const
```

Obtiene el nombre del día de la semana por medio de un índice.

```
string  DayName(
   const int      num         // índice del día
   )  const
```

Parámetros

num

[in] Índice del día (0-6).

Valor devuelto

Nombre del día.

© 2000-2025, MetaQuotes Ltd.

ShortDayName

Obtiene el nombre corto del día de la semana.

```
string  ShortDayName()  const
```

Obtiene el nombre corto del día de la semana por medio de un índice.

```
string  ShortDayName(
   const int      num         // índice del día
   )  const
```

Parámetros

num

[in] Índice del día (0-6).

Valor devuelto

Nombre corto del día.

© 2000-2025, MetaQuotes Ltd.

DaysInMonth

Obtiene el número de días del mes.

```
int  DaysInMonth()  const
```

Valor devuelto

Número de días del mes.

© 2000-2025, MetaQuotes Ltd.

DateTime (Método Get)

Obtiene la fecha y la hora.

```
datetime  DateTime()
```

Valor devuelto

Valor del tipo datetime.

DateTime (Método Set datetime)

Establece la fecha y la hora con el tipo datetime.

```
void  DateTime(
   const datetime      value          // fecha y hora
   )
```

Parámetros

value

[in] Valor del tipo datetime.

Valor devuelto

Ninguno.

DateTime (Método Set MqlDateTime)

Establece la fecha y la hora con el tipo MqlDateTime.

```
void  DateTime(
   const MqlDateTime   &value         // fecha y hora
   )
```

Parámetros

value

[in] Valor del tipo MqlDateTime.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Date (Método Set datetime)

Establece la fecha con el tipo datetime.

```
void  Date(
   const datetime      value         // fecha
   )
```

Parámetros

value

[in] Valor del tipo datetime.

Valor devuelto

Ninguno.

Date (Método Set MqlDateTime)

Establece la fecha con el tipo MqlDateTime.

```
void  Date(
   const MqlDateTime   &value        // fecha
   )
```

Parámetros

value

[in] Valor del tipo MqlDateTime.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Time (Método Set datetime)

Establece la hora con el tipo datetime.

```
void  Time(
   const datetime      value          // hora
   )
```

Parámetros

value

[in] Valor del tipo datetime.

Valor devuelto

Ninguno.

Time (Método Set MqlDateTime)

Establece la hora con el tipo MqlDateTime.

```
void  Time(
   const MqlDateTime   &value         // hora
   )
```

Parámetros

value

[in] Valor del tipo MqlDateTime.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Sec

Establece los segundos.

```
void  Sec(
   const int   value        // segundos
   )
```

Parámetros

value

[in] Segundos.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Min

Establece los minutos.

```
void  Min(
   const int    value        // minutos
   )
```

Parámetros

value

[in] Minutos.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Hour

Establece la hora.

```
void  Hour(
   const int    value        // hora
   )
```

Parámetros

value

[in] Hora.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Day

Establece el día del mes.

```
void  Day(
   const int    value        // día
   )
```

Parámetros

value

[in] Día del mes.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Mon

Establece el mes.

```
void  Mon(
   const int    value        // mes
   )
```

Parámetros

value

[in] Mes.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Year

Establece el año.

```
void  Year(
   const int    value        // año
   )
```

Parámetros

value

[in] Año.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

SecDec

Resta el número de segundos especificado.

```
void  SecDec(
   int   delta=1        // segundos
   )
```

Parámetros

delta

[in] Segundos a sustraer.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

SecInc

Añade el número de segundos especificado.

```
void  SecInc(
   int   delta=1        // segundos
   )
```

Parámetros

delta

[in] Segundos a añadir.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

MinDec

Resta el número de minutos especificado.

```
void  MinDec(
   int   delta=1        // minutos
   )
```

Parámetros

delta

[in] Minutos a sustraer.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

MinInc

Añade el número de minutos especificado.

```
void  MinInc(
   int   delta=1        // minutos
   )
```

Parámetros

delta

[in] Minutos a añadir.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

HourDec

Resta el número de horas especificado.

```
void  HourDec(
   int   delta=1        // horas
   )
```

Parámetros

delta

[in] Horas a sustraer.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

HourInc

Añade el número de horas especificado.

```
void  HourInc(
   int   delta=1        // horas
   )
```

Parámetros

delta

[in] Horas a añadir.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

DayDec

Resta el número de días especificado.

```
void  DayDec(
   int   delta=1        // días
   )
```

Parámetros

delta

[in] Días a sustraer.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

DayInc

Añade el número de días especificado.

```
void  DayInc(
   int   delta=1        // días
   )
```

Parámetros

delta

[in] Días a añadir.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

MonDec

Subtracts specified number of days.

```
void  MonDec(
   int   delta=1        // months
   )
```

Parameters

delta

[in] Months to subtract.

Returned value

None.

© 2000-2025, MetaQuotes Ltd.

MonInc

Adds specified number of days.

```
void  MonInc(
   int   delta=1        // months
   )
```

Parameters

delta

[in] Months to add.

Returned value

None.

© 2000-2025, MetaQuotes Ltd.

YearDec

Resta el número de años especificado.

```
void  YearDec(
   int   delta=1        // años
   )
```

Parámetros

delta

[in] Años a sustraer.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

YearInc

Añade el número de años especificado.

```
void  YearInc(
   int   delta=1        // años
   )
```

Parámetros

delta

[in] Años a añadir.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

CWnd

CWnd es una clase base de todos los controles, incluida en la Librería Estándar MQL5.

Descripción

La clase CWnd es la implementación de la clase base de control.

Declaración

```
class CWnd : public CObject
```

Título

```
#include <Controls\Wnd.mqh>
```

Jerarquía de herencia

CObject

CWnd

Descendientes directos

CDragWnd, CWndContainer, CWndObj

Métodos de la clase

Crear y destruir	
Create	Crea el control
Destroy	Destruye el control
Manejadores de eventos gráficos	
OnEvent	Manejador de todos los eventos gráficos
OnMouseEvent	Manejador del evento CHARTEVENT_MOUSE_MOVE
Nombre	
Name	Obtiene el nombre del control
Acceso al contenedor	
ControlsTotal	Obtiene el número de controles del contenedor
Control	Obtiene el control por medio del índice
ControlFind	Obtiene el control por medio del ID
Geometría	
Rect	Obtiene el puntero al objeto de la clase CRect

© 2000-2025, MetaQuotes Ltd.

Crear y destruir	
Left	Obtiene/Establece la coordenada X de la esquina superior izquierda
Top	Obtiene/Establece la coordenada Y de la esquina superior izquierda
Right	Obtiene/Establece la coordenada X de la esquina inferior derecha
Bottom	Obtiene/Establece la coordenada Y de la esquina inferior derecha
Width	Obtiene/Establece la anchura
Height	Obtiene/Establece la altura
Move	Establece las nuevas coordenadas del control
Shift	Realiza el desplazamiento relativo de las coordenadas del control
Resize	Establece la nueva anchura/altura del control
Contains	Comprueba si el punto/control está dentro del área de control
Alineación	
Alignment	Establece las propiedades de alineación del control
Align	Realiza la alineación del control
Identificación	
Id	Obtiene/Establece el ID del control
Estado	
IsEnabled	Obtiene un valor que indica si el control está habilitado
Enable	Establece un valor que indica que el control está habilitado
Disable	Inhabilita el control
IsVisible	Comprueba la bandera de visibilidad
Visible	Establece la bandera de visibilidad
Show	Muestra el control
Hide	Esconde el control
IsActive	Comprueba la actividad del control
Activate	Activa el control
Deactivate	Desactiva el control
Banderas de estado	
StateFlags	Obtiene/Establece las banderas de estado del control
StateFlagsSet	Establece las banderas de estado del control

© 2000-2025, MetaQuotes Ltd.

Crear y destruir	
StateFlagsReset	Reinicia las banderas de estado del control
Banderas de propiedades	
PropFlags	Obtiene/Establece las banderas de propiedades del control
PropFlagsSet	Establece las banderas de propiedades del control
PropFlagsReset	Reinicia las banderas de propiedades del control
Operaciones de ratón	
MouseX	Obtiene/Establece la coordenada X del ratón
MouseY	Obtiene/Establece la coordenada Y del ratón
MouseFlags	Obtiene/Establece el estado de los botones del ratón
MouseFocusKill	Finaliza el foco del ratón
Manejadores de evento internos	
OnCreate	Manejador de evento "Crear"
OnDestroy	Manejador del evento "Destruir"
OnMove	Manejador de evento "Mover"
OnResize	Manejador de evento "Cambiar tamaño"
OnEnable	Manejador del evento "Activar"
OnDisable	Manejador del evento "Desactivar"
OnShow	Manejador de evento "Mostrar"
OnHide	Manejador de evento "Esconder"
OnActivate	Manejador de evento "Activar"
OnDeactivate	Manejador de evento "Desactivar"
OnClick	Manejador de evento "Clicar"
OnChange	Manejador del evento "Cambiar"
Manejadores de eventos de ratón	
OnMouseDown	Manejador de evento "Ratón abajo"
OnMouseUp	Manejador de eveno "Ratón arriba"
Manejadores de eventos de arrastre	
OnDragStart	Manejador del evento "Inicio de arrastre"

© 2000-2025, MetaQuotes Ltd.

Crear y destruir	
OnDragProcess	Manejador del evento "Proceso de arrastre"
OnDragEnd	Manejador del evento "Finalizar arrastre"
Objeto arrastrar	
DragObjectCreate	Crea el objeto de arrastre
DragObjectDestroy	Destruye el objeto de arrastre

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Create

Crea el control.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

El método de la clase base solo guarda los parámetros y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

Destroy

Destruye el control.

```
virtual bool  Destroy()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnMouseEvent

Manejador de evento de ratón (el evento gráfico CHARTEVENT_MOUSE_MOVE).

```
virtual bool  OnMouseEvent(
   const int  x,          // coordenada x
   const int  y,          // coordenada y
   const int  flags       // banderas
   )
```

Parámetros

x

[in] Coordenada X del cursor del ratón relativa a la esquina superior izquierda del gráfico.

y

[in] Coordenada Y del cursor del ratón relativa a la esquina superior izquierda del gráfico.

flags

[in] Bandera de estados de los botones del ratón.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Name

Obtiene el nombre del control.

```
string  Name()  const
```

Valor devuelto

Nombre del control.

© 2000-2025, MetaQuotes Ltd.

ControlsTotal

Obtiene el número de controles del contenedor.

```
int  ControlsTotal()  const
```

Valor devuelto

Número de controles del contenedor.

Nota

El método de la clase base no incluye el contenedor, sino que proporciona a sus hijos acceso al mismo, y siempre devuelve 0.

© 2000-2025, MetaQuotes Ltd.

Control

Obtiene el control por medio del índice.

```
CWnd*  Control(
   const int  ind      // índice
   )  const
```

Parámetros

ind

[in] Índice del control.

Valor devuelto

Un puntero al control.

Nota

El método de la clase base no incluye el contenedor, sino que proporciona a sus hijos acceso al mismo, y siempre devuelve NULL.

© 2000-2025, MetaQuotes Ltd.

ControlFind

Obtiene el control del contenedor por medio del ID especificado.

```
virtual CWnd*  ControlFind(
   const long  id      // ID
   )
```

Parámetros

id

[in] Identificador del control a encontrar.

Valor devuelto

Puntero al control.

Nota

El método de la clase base no incluye el contenedor, sino que proporciona a sus hijos acceso al mismo. Si el identificador especificado coincide con el ID del contenedor, devuelve un puntero a sí mismo (this).

© 2000-2025, MetaQuotes Ltd.

Rect

Obtiene el puntero al objeto de la clase CRect.

```
const CRect*  Rect()  const
```

Valor devuelto

Puntero al objeto de la clase CRect.

© 2000-2025, MetaQuotes Ltd.

Left (Método Get)

Obtiene la coordenada X de la esquina superior izquierda del control.

```
int  Left()
```

Valor devuelto

La coordenada X de la esquina superior izquierda del control.

Left (Método Set)

Establece la coordenada X de la esquina superior izquierda del control.

```
void  Left(
   const int  x      // nueva coordenada x
   )
```

Parámetros

x

[in] Nueva coordenada X de la esquina superior izquierda.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Top (Método Get)

Obtiene la coordenada Y de la esquina superior izquierda del control.

```
int  Top()
```

Valor devuelto

La coordenada Y de la esquina superior izquierda del control.

Top (Método Set)

Establece la coordenada Y de la esquina superior izquierda del control.

```
void  Top(
   const int  y      // coordenada y
   )
```

Parámetros

y

[in] Nueva coordenada Y de la esquina superior izquierda.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Right (Método Get)

Obtiene la coordenada X de la esquina inferior derecha del control.

```
int  Right()
```

Valor devuelto

Coordenada X de la esquina inferior derecha.

Right (Método Set)

Establece la coordenada Y de la esquina inferior derecha del control.

```
void  Right(
   const int  x      // coordenada x
   )
```

Parámetros

x

[in] Nueva coordenada X de la esquina inferior derecha.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Bottom (Método Get)

Obtiene la coordenada Y de la esquina inferior derecha del control.

```
int  Bottom()
```

Valor devuelto

Coordenada Y de la esquina inferior derecha del control.

Bottom (Método Set)

Establece la coordenada Y de la esquina inferior derecha del control.

```
void  Bottom(
   const int  y      // coordenada y
   )
```

Parámetros

y

[in] Nueva coordenada Y de la esquina inferior derecha.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Width (Método Get)

Obtiene la anchura del control.

```
int  Width()
```

Valor devuelto

Anchura del control.

Width (Método Set)

Establece la nueva anchura del control.

```
virtual bool  Width(
   const int  w      // anchura
   )
```

Parámetros

w

[in] Nueva anchura.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Height (Método Get)

Obtiene la altura del control.

```
int  Height()
```

Valor devuelto

Altura del control.

Height (Método Set)

Establece la nueva altura del control.

```
virtual bool  Height(
   const int  h      // altura
   )
```

Parámetros

h

[in] Nueva altura.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Move

Establece las nuevas coordenadas del control.

```
void  Move(
   const int  x,      // coordenada X
   const int  y       // coordenada Y
   )
```

Parámetros

x

[in] Nueva coordenada X.

y

[in] Nueva coordenada Y.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Shift

Realiza el desplazamiento relativo de las coordenadas del control.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Parámetros

dx

[in] Delta X.

dy

[in] Delta Y.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Resize

Establece la nueva anchura/altura del control.

```
virtual bool  Resize(
   const int  w,      // anchura
   const int  h       // altura
   )
```

Parámetros

w

[in] Nueva anchura.

h

[in] Nueva altura.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Contains

Comprueba si el punto está dentro del área de control del gráfico.

```
bool  Contains(
   const int  x,      // coordenada X
   const int  y       // coordenada Y
   )
```

Parámetros

x

[in] coordenada X.

y

[in] coordenada Y.

Valor devuelto

true, si el punto está dentro del área (incluyendo los bordes), false en caso contrario.

Contains

Comprueba si el control especificado está dentro del área de control del gráfico.

```
bool  Contains(
   const CWnd*  control      // puntero
   )  const
```

Parámetros

control

[in] Puntero al objeto.

Valor devuelto

true, si el control especificado está dentro del área (incluyendo los bordes), false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Alignment

Establece los parámetros de alineación del control.

```
void  Alignment(
   const int  flags,       // banderas de alineación
   const int  left,        // desplazamiento desde el borde izquierdo
   const int  top,         // desplazamiento desde el borde superior
   const int  right,       // desplazamiento desde el borde derecho
   const int  bottom       // desplazamiento desde el borde inferior
   )
```

Parámetros

flags

[in] Banderas de alineación.

left

[in] Desplazamiento fijo desde el borde izquierdo.

top

[in] Desplazamiento fijo desde el borde superior.

right

[in] Desplazamiento fijo desde el borde derecho.

bottom

[in] Desplazamiento fijo desde el borde inferior.

Valor devuelto

Ninguno.

Nota

Banderas de alineación:

```
enum WND_ALIGN_FLAGS
{
   WND_ALIGN_NONE=0,                                  // sin alineación
   WND_ALIGN_LEFT=1,                                  // alineación izquierda
   WND_ALIGN_TOP=2,                                   // alineación superior
   WND_ALIGN_RIGHT=4,                                 // alineación derecha
   WND_ALIGN_BOTTOM=8,                                // alineación inferior
   WND_ALIGN_WIDTH = WND_ALIGN_LEFT|WND_ALIGN_RIGHT,  // ancho de la alineación
   WND_ALIGN_HEIGHT=WND_ALIGN_TOP|WND_ALIGN_BOTTOM,   // altura de la alineación
   WND_ALIGN_CLIENT=WND_ALIGN_WIDTH|WND_ALIGN_HEIGHT, // altura y anchura de la alinea
   }
```

© 2000-2025, MetaQuotes Ltd.

Align

Realiza la alineación del control en el área gráfica especificada.

```
virtual bool  Align(
   const CRect*  rect      // puntero
   )
```

Parámetros

rect

[in] Puntero al objeto con las coordenadas del área gráfica.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

Los parámetros de alineación se tienen que especificar (no hay ninguna alineación predeterminada).

© 2000-2025, MetaQuotes Ltd.

Id (Método Get)

Obtiene el ID del control.

```
long  Id()  const
```

Valor devuelto

El identificador del control.

Id (Método Set)

Establece el nuevo valor del identificador del control.

```
virtual long  Id(
   const long  id      // identificador
   )
```

Parámetros

id

[in] Valor nuevo del identificador del control.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

IsEnabled

Obtiene un valor que indica si el control está habilitado.

```
bool  IsEnabled()  const
```

Valor devuelto

true - si el control está habilitado, en caso contrario - false.

© 2000-2025, MetaQuotes Ltd.

Enable

Activa el control.

```
virtual bool  Enable()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

Si el control está habilitado, puede procesar eventos externos.

© 2000-2025, MetaQuotes Ltd.

Disable

Desactiva el control.

```
virtual bool  Disable()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

El control deshabilitado no puede procesar los eventos externos.

© 2000-2025, MetaQuotes Ltd.

IsVisible

Obtiene un valor que indica si el control es visible.

```
bool  IsVisible()  const
```

Valor devuelto

true si el control se muestra en el gráfico, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Visible

Establece la bandera de visibilidad.

```
virtual bool  Visible(
   const bool  flag      // bandera
   )
```

Parámetros

flag

[in] Nueva bandera.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Show

Muestra el control.

```
virtual bool  Show()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Hide

Esconde el control.

```
virtual bool  Hide()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

IsActive

Obtiene un valor que indica si el control está activo.

```
bool  IsActive()  const
```

Valor devuelto

true si el control está activo, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Activate

Activa el control.

```
virtual bool  Activate()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

El control se activa al pasar por encima el cursor del ratón.

© 2000-2025, MetaQuotes Ltd.

Deactivate

Desactiva el control.

```
virtual bool  Deactivate()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

El control se vuelve inactivo cuando el cursor del ratón está fuera de control.

© 2000-2025, MetaQuotes Ltd.

StateFlags (Método Get)

Obtiene las banderas de estado del control.

```
int  StateFlags()
```

Valor devuelto

Las banderas de estado del control.

StateFlags (Método Set)

Establece las banderas de estado del control.

```
virtual void  StateFlags(
   const int  flags      // banderas
   )
```

Parámetros

flags

[in] Nuevas banderas de estado del control.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

StateFlagsSet

Establece las banderas de estado del control.

```
virtual void  StateFlagsSet(
   const int  flags      // banderas
   )
```

Parámetros

flags

[in] Banderas a establecer (máscara de bits).

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

StateFlagsReset

Reinicia las banderas de estado del control.

```
virtual void  StateFlagsReset(
   const int  flags      // banderas
   )
```

Parámetros

flags

[in] Banderas a reiniciar (máscara de bits).

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

PropFlags (Método Get)

Obtiene las banderas de propiedades del control.

```
void  PropFlags(
   const int  flags      // banderas
   )
```

Valor devuelto

Las banderas de propiedades del control.

PropFlags (Método Set)

Establece las banderas de propiedades del control.

```
virtual void  PropFlags(
   const int  flags      // banderas
   )
```

Parámetros

flags

[in] Banderas nuevas.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

PropFlagsSet

Establece las banderas de propiedades del control.

```
virtual void  PropFlagsSet(
   const int  flags      // banderas
   )
```

Parámetros

flags

[in] Banderas a establecer (máscara de bits).

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

PropFlagsReset

Reinicia las banderas de propiedades del control.

```
virtual void  PropFlagsReset(
   const int  flags      // banderas
   )
```

Parámetros

flags

[in] Banderas a reiniciar (máscara de bits).

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

MouseX (Método Set)

Guarda la coordenada X del ratón.

```
void  MouseX(
   const int  value      // coordenada
   )
```

Parámetros

value

[in] La coordenada X del ratón.

Valor devuelto

Ninguno.

MouseX (Método Get)

Obtiene la coordenada X del ratón guardada.

```
int  MouseX()
```

Valor devuelto

La coordenada X del ratón guardada.

© 2000-2025, MetaQuotes Ltd.

MouseY (Método Set)

Guarda la coordenada Y del ratón.

```
void  MouseY(
   const int  value      // coordenada
   )
```

Parámetros

value

[in] La coordenada Y del ratón.

Valor devuelto

Ninguno.

MouseY (Método Get)

Obtiene la coordenada Y del ratón guardada.

```
int  MouseY()
```

Valor devuelto

La coordenada Y del ratón guardada.

© 2000-2025, MetaQuotes Ltd.

MouseFlags (Método Set)

Guarda el estado de los botones del ratón.

```
virtual void  MouseFlags(
   const int  value      // estado
   )
```

Parámetros

value

[in] Estado de los botones del ratón.

Valor devuelto

Ninguno.

MouseFlags (Método Get)

Obtiene el estado guardado de los botones del ratón.

```
int  MouseFlags()
```

Valor devuelto

Estado de los botones del ratón.

© 2000-2025, MetaQuotes Ltd.

MouseFocusKill

Borra el estado guardado de los botones del ratón y desactiva el control.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // id
   )
```

Parámetros

id=CONTROLS_INVALID_ID

[in] Identificador del control que recibe el foco del ratón.

Valor devuelto

El resultado de la desactivación del control.

© 2000-2025, MetaQuotes Ltd.

OnCreate

El manejador de evento del control "Crear".

```
virtual bool  OnCreate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnDestroy

El manejador de evento del control "Destruir".

```
virtual bool  OnDestroy()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnMove

El manejador de evento del control "Mover".

```
virtual bool  OnMove()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnResize

El manejador de evento del control "Cambiar tamaño".

```
virtual bool  OnResize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnEnable

El manejador de evento del control "Habilitar" (si está habilitado, puede responder a la interacción del usuario).

```
virtual bool  OnEnable()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnDisable

El manejador de evento del control "Deshabilitar" (si está deshabilitado no puede responder a la interacción del usuario).

```
virtual bool  OnDisable()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnShow

El manejador de evento del control "Mostrar".

```
virtual bool  OnShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnHide

El manejador de evento del control "Esconder".

```
virtual bool  OnHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnActivate

El manejador de evento del control "Activar".

```
virtual bool  OnActivate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnDeactivate

El manejador de evento del control "Desactivar".

```
virtual bool  OnDeactivate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnClick

El manejador de evento del control "Clicar" (clic con el botón izquierdo del ratón).

```
virtual bool  OnClick()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChange

El manejador de evento del control "Cambiar".

```
virtual bool  OnChange()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnMouseDown

El manejador de evento del control "Ratón abajo".

```
virtual bool  OnMouseDown()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Ratón abajo" ocurre al hacer clic con el botón izquierdo del ratón sobre el control.

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

El manejador de evento del control "Ratón arriba" (se suelta el botón izquierdo del ratón).

```
virtual bool  OnMouseUp()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Ratón arriba" ocurre cuando se suelta el botón izquierdo del ratón sobre el control.

© 2000-2025, MetaQuotes Ltd.

OnDragStart

El manejador de evento del control "Empezar arrastre".

```
virtual bool  OnDragStart()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Empezar arrastre" se produce cuando comienza la operación de arrastrar.

© 2000-2025, MetaQuotes Ltd.

OnDragProcess

El manejador de evento del control "Procesar arrastre".

```
virtual bool  OnDragProcess(
   const int  x,      // coordenada x
   const int  y       // coordenada y
   )
```

Parámetros

x

[in] Coordenada X del cursor del ratón.

y

[in] Coordenada Y actual del puntero del ratón.

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Procesar arrastre" se produce al mover el control.

© 2000-2025, MetaQuotes Ltd.

OnDragEnd

El manejador de evento del control "Finalizar arrastre".

```
virtual bool  OnDragEnd()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Finalizar arrastre" se produce cuando finaliza el proceso de arrastre del control.

© 2000-2025, MetaQuotes Ltd.

DragObjectCreate

Crea el objeto de arrastre.

```
virtual bool  DragObjectCreate()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

DragObjectDestroy

Destruye el objeto de arrastre.

```
virtual bool  DragObjectDestroy()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CWndObj

CWndObj es una clase base de controles simples (basados en objetos gráficos) de la Librería Estándar.

Descripción

La clase CWndObj implementa métodos base de control simple.

Declaración

```
class CWndObj : public CWnd
```

Título

```
#include <Controls\WndObj.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndObj

Descendientes directos

CBmpButton, CButton, CEdit, CLabel, CPanel, CPicture

Métodos de la clase

Procesamiento de eventos gráficos	
OnEvent	Manejador de todos los eventos gráficos
Propiedades	
Text	Obtiene/Establece la propiedad OBJPROP_TEXT del objeto gráfico
Color	Obtiene/Establece la propiedad OBJPROP_COLOR del objeto gráfico
ColorBackground	Obtiene/Establece la propiedad OBJPROP_BGCOLOR del objeto gráfico
ColorBorder	Obtiene/Establece la propiedad OBJPROP_BORDER_COLOR del objeto gráfico
Font	Obtiene/Establece la propiedad OBJPROP_FONT del objeto gráfico
FontSize	Obtiene/Establece la propiedad OBJPROP_FONTSIZE del objeto gráfico
ZOrder	Obtiene/Establece la propiedad OBJPROP_ZORDER del objeto gráfico

© 2000-2025, MetaQuotes Ltd.

Procesamiento de eventos gráficos	
Manejadores de eventos de objetos gráficos	
OnObjectCreate	Manejador de evento CHARTEVENT_OBJECT_CREATE
OnObjectChange	Manejador de evento CHARTEVENT_OBJECT_CHANGE
OnObjectDelete	Manejador de evento CHARTEVENT_OBJECT_DELETE
OnObjectDrag	Manejador de evento CHARTEVENT_OBJECT_DRAG
Manejadores de evento de cambio de propiedad	
OnSetText	Manejador del evento "Establecer texto"
OnSetColor	Manejador del evento "Establecer color"
OnSetColorBackground	Manejador del evento "Establecer color de fondo"
OnSetFont	Manejador del evento "Establecer fuente"
OnSetFontSize	Manejador del evento "Establecer tamaño de la fuente"
OnSetZOrder	Manejador del evento "Establecer orden Z"
Manejadores de evento internos	
OnDestroy	Manejador del evento "Destruir"
OnChange	Manejador del evento "Cambiar"

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CWnd

Create, Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Text (Método Get)

Obtiene la propiedad OBJPROP_TEXT (texto) del objeto gráfico.

```
string  Text()
```

Valor devuelto

El valor de la propiedad OBJPROP_TEXT.

Text (Método Set)

Establece la propiedad OBJPROP_TEXT (texto) del objeto gráfico.

```
bool  Text(
   const string  value      // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo de la propiedad OBJPROP_TEXT.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Color (Método Get)

Obtiene la propiedad OBJPROP_COLOR (color) del objeto gráfico.

```
color  Color()
```

Valor devuelto

El valor de la propiedad OBJPROP_COLOR.

Color (Método Set)

Establece la propiedad OBJPROP_COLOR (color) del objeto gráfico.

```
bool  Color(
   const color  value      // valor
   )
```

Parámetros

value

[in] Valor nuevo de la propiedad OBJPROP_COLOR.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ColorBackground (Método Get)

Obtiene la propiedad OBJPROP_BGCOLOR (color de fondo) del objeto gráfico.

```
color  ColorBackground()
```

Valor devuelto

El valor de la propiedad OBJPROP_BGCOLOR.

ColorBackground (Método Set)

Establece la propiedad OBJPROP_BGCOLOR (color de fondo) del objeto gráfico.

```
bool  ColorBackground(
   const color  value      // valor
   )
```

Parámetros

value

[in] Valor nuevo de la propiedad OBJPROP_BGCOLOR.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ColorBorder (Método Get)

Obtiene la propiedad OBJPROP_BORDER_COLOR (color de borde) del objeto gráfico.

```
color  ColorBorder()
```

Valor devuelto

El valor de la propiedad OBJPROP_BORDER_COLOR.

ColorBorder (Método Set)

Establece la propiedad OBJPROP_BORDER_COLOR (color de borde) del objeto gráfico.

```
bool  ColorBorder(
   const color  value      // valor
   )
```

Parámetros

value

[in] Valor nuevo de la propiedad OBJPROP_BORDER_COLOR.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Font (Método Get)

Obtiene la propiedad OBJPROP_FONT (fuente) del objeto gráfico.

```
string  Font()
```

Valor devuelto

El valor de la propiedad OBJPROP_FONT.

Font (Método Set)

Establece la propiedad OBJPROP_FONT (fuente) del objeto gráfico.

```
bool  Font(
   const string  value      // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo de la propiedad OBJPROP_FONT.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

FontSize (Método Get)

Obtiene la propiedad OBJPROP_FONTSIZE (tamaño de la fuente) del objeto gráfico.

```
int  FontSize()
```

Valor devuelto

El valor de la propiedad OBJPROP_FONTSIZE.

FontSize (Método Set)

Establece la propiedad OBJPROP_FONTSIZE (tamaño de la fuente) del objeto gráfico.

```
bool  FontSize(
   const int  value      // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo de la propiedad OBJPROP_FONTSIZE.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ZOrder (Método Get)

Obtiene la propiedad OBJPROP_ZORDER del objeto gráfico.

```
long  ZOrder()
```

Valor devuelto

El valor de la propiedad OBJPROP_ZORDER.

ZOrder (Método Set)

Establece la propiedad OBJPROP_ZORDER del objeto gráfico.

```
bool  ZOrder(
   const long  value      // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo de la propiedad OBJPROP_ZORDER.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnObjectCreate

El manejador de evento de objeto gráfico CHARTEVENT_OBJECT_CREATE.

```
virtual bool  OnObjectCreate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnObjectChange

El manejador de evento de objeto gráfico CHARTEVENT_OBJECT_CHANGE.

```
virtual bool  OnObjectChange()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnObjectDelete

El manejador de evento de objeto gráfico CHARTEVENT_OBJECT_DELETE.

```
virtual bool  OnObjectDelete()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnObjectDrag

>El manejador de evento de objeto gráfico CHARTEVENT_OBJECT_DRAG.

```
virtual bool  OnObjectDrag()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetText

El manejador de evento del control "Establecer texto" (cambio de la propiedad OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

El manejador de evento "Establecer color" (cambio de la propiedad OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

El manejador de evento del control "Establecer color de fondo" (cambio de la propiedad OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

El manejador de evento del control "Establecer fuente" (cambio de la propiedad OBJPROP_FONT).

```
virtual bool  OnSetFont()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

El manejador de evento "Establecer tamaño de fuente" (cambio de la propiedad OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

El manejador de evento del control "Establecer orden Z" (cambio de la propiedad OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnDestroy

El manejador de evento del control "Destruir".

```
virtual bool  OnDestroy()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChange

El manejador de evento del control "Cambiar".

```
virtual bool  OnChange()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CWndContainer

CWndContainer es la clase base de un control complejo (con controles dependientes) de la Librería Estándar.

Descripción

La clase CWndContainer implementa métodos base del control complejo.

Declaración

```
class CWndContainer : public CWnd
```

Título

```
#include <Controls\WndContainer.mqh>
```

Jerarquía de herencia

CObject

CWnd

CWndContainer

Descendientes directos

CCheckBox, CComboBox, CDateDropList, CDatePicker, CDialog, CRadioButton, CScroll, CSpinEdit, CWndClient

Métodos de la clase

Destruir	
Destroy	Destruye todos los controles del contenedor
Manejadores de eventos gráficos	
OnEvent	Manejador de todos los eventos gráficos
OnMouseEvent	El manejador de evento CHARTEVENT_MOUSE_MOVE
Acceso al contenedor	
ControlsTotal	Obtiene el número de controles del contenedor
Control	Obtiene el control por medio del índice
ControlFind	Obtiene el control por medio del identificador
Añadir/Borrar	
Add	Añade el control al contenedor
Delete	Borra el control del contenedor
Geometría	

© 2000-2025, MetaQuotes Ltd.

Destruir	
Move	Establece las nuevas coordenadas de todos los controles del contenedor
Shift	Realiza el desplazamiento relativo de las coordenadas de todos los controles del contenedor
Identificación	
Id	Establece el ID de todos los controles del contenedor
Estado	
Enable	Habilita todos los controles del contenedor
Disable	Deshabilita todos los controles del contenedor
Show	Muestra todos los controles del contenedor
Hide	Esconde todos los controles del contenedor
Operaciones de ratón	
MouseFocusKill	Finaliza el foco del ratón
Operaciones con archivos	
Save	Guarda la información del contenedor en el archivo
Load	Carga la información del contenedor a partir del archivo
Manejadores de evento internos	
OnResize	Manejador de evento "Cambiar tamaño"
OnActivate	Manejador de evento "Activar"
OnDeactivate	Manejador de evento "Desactivar"

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Create, Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2025, MetaQuotes Ltd.

Destroy

Destruye todos los controles del contenedor.

```
virtual bool  Destroy()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnMouseEvent

Manejador de evento del ratón.

```
virtual bool  OnMouseEvent(
   const int  x,          // coordenada x
   const int  y,          // coordenada y
   const int  flags       // banderas
   )
```

Parámetros

x

[in] Coordenada X del cursor del ratón relativa a la esquina superior izquierda del gráfico.

y

[in] Coordenada Y del cursor del ratón relativa a la esquina superior izquierda del gráfico.

flags

[in] Bandera de estados de los botones del ratón.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ControlsTotal

Obtiene el número de controles del contenedor.

```
int  ControlsTotal()  const
```

Valor devuelto

Número de controles del contenedor.

© 2000-2025, MetaQuotes Ltd.

Control

Obtiene el control del contenedor por medio del índice.

```
CWnd*  Control(
   const int  ind      // índice
   )  const
```

Parámetros

ind

[in] Índice del control.

Valor devuelto

Puntero al control, en caso contrario NULL si el control no se encuentra.

© 2000-2025, MetaQuotes Ltd.

ControlFind

Obtiene el control del contenedor por medio del identificador.

```
virtual CWnd*  ControlFind(
   const long  id      // id
   )
```

Parámetros

id

[in] Identificador del control.

Valor devuelto

Puntero al control, en caso contrario NULL si el control no se encuentra.

© 2000-2025, MetaQuotes Ltd.

Add

Añade el control al contenedor.

```
bool  Add(
   CWnd&  control      // referencia
   )
```

Parámetros

control

[in] Control a añadir, pasado por referencia.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Delete

Borra el control del contenedor.

```
bool  Delete(
   CWnd&  control      // referencia
   )
```

Parámetros

control

[in] Control a borrar, pasado por referencia.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Move

Establece las nuevas coordenadas de todos los controles del contenedor.

```
virtual bool  Move(
   const int  x,      // coordenada x
   const int  y       // coordenada y
   )
```

Parámetros

x

[in] Nueva coordenada X de la esquina superior izquierda.

y

[in] Nueva coordenada Y de la esquina superior izquierda.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Shift

Realiza el desplazamiento relativo de las coordenadas de todos los controles del contenedor.

```
virtual bool  Shift(
   const int  dx,      // delta x
   const int  dy       // delta y
   )
```

Parámetros

dx

[in] Delta X.

dy

[in] Delta Y.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Id

Establece el identificador de todos los controles del contenedor.

```
virtual long  Id(
   const long  id      // identificador
   )
```

Parámetros

id

[in] Identificador del grupo base.

Valor devuelto

Número de identificadores utilizado por los controles del contenedor.

© 2000-2025, MetaQuotes Ltd.

Enable

Habilita todos los controles del contenedor.

```
virtual bool  Enable()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Disable

Deshabilita todos los controles del contenedor.

```
virtual bool  Disable()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Show

Muestra todos los controles del contenedor.

```
virtual bool  Show()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Hide

Esconde todos los controles del contenedor.

```
virtual bool  Hide()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

MouseFocusKill

Reinicia el estado guardado de los botones del ratón y desactiva todos los controles del contenedor.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // id
   )
```

Parámetros

id=CONTROLS_INVALID_ID

[in] Identificador del control que recibe el foco del ratón.

Valor devuelto

El resultado de desactivar los controles.

© 2000-2025, MetaQuotes Ltd.

Save

Guarda la información del contenedor en el archivo.

```
virtual bool  Save(
   const int  file_handle      // manejador
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario (debe abrirse para escritura).

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Load

Carga la información del contenedor a partir del archivo

```
virtual bool  Load(
   const int  file_handle      // manejador
   )
```

Parámetros

file_handle

[in] Manejador del archivo binario (tiene que abrirse para lectura).

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnResize

El manejador de evento del control "Cambiar tamaño".

```
virtual bool  OnResize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnActivate

El manejador de evento del control "Activar".

```
virtual bool  OnActivate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnDeactivate

El manejador de evento del control "Desactivar".

```
virtual bool  OnDeactivate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

Klasse CLabel

Klasse CLabel ist eine Klasse des einfachen Steuerelements basierend auf das grafischen Objekt "Text-Label".

Beschreibung

Klasse CLabel wird für die Erstellung von einfachen nicht-bearbeitbaren Textfeldern verwendet.

Deklaration

```
class CLabel : public CWndObj
```

Kopf

```
#include <Controls\Label.mqh>
```

Jerarquía de herencia

CObject
CWnd
CWndObj
CLabel

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

Erstellung	
Create	Erstellt ein Steuerelement

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Parameteränderungenbehandlung	
OnSetText	Handler des Ereignisses "SetText" eines Steuerelements
OnSetColor	Handler des Ereignisses "SetColor" eines Steuerelements
OnSetFont	Handler des Ereignisses "SetFont" eines Steuerelements
OnSetFontSize	Handler des Ereignisses "SetFontSize" eines Steuerelements
Behandlung von internen Ereignissen	
OnCreate	Handler des internen Ereignisses "Create" eines Steuerelements
OnShow	Handler des internen Ereignisses "Show" eines Steuerelements
OnHide	Handler des internen Ereignisses "Hide" eines Steuerelements
OnMove	Handler des internen Ereignisses "Move" eines Steuerelements

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Ein Beispiel für die Erstellung eines Panels mit einem Text-Label:

```
//+------------------------------------------------------------------+
//|                                                ControlsLabel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CLabel"
#include <Controls\Dialog.mqh>
#include <Controls\Label.mqh>
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CLabel            m_label;                        // CLabel object
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
protected:
   //--- create dependent controls
   bool              CreateLabel(void);
   //--- handlers of the dependent controls events
   void              OnClickLabel(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)

EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateLabel())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CLabel"                                              |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateLabel(void)
  {
//--- coordinates
   int x1=INDENT_RIGHT;
   int y1=INDENT_TOP+CONTROLS_GAP_Y;
   int x2=x1+100;
   int y2=y1+20;
//--- create
   if(!m_label.Create(m_chart_id,m_name+"Label",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_label.Text("Label"))
      return(false);
   if(!Add(m_label))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CLabel.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetText

El manejador de evento del control "Establecer texto" (cambio de la propiedad OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

El manejador de evento del control "Establecer color" (cambio de la propiedad OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

El manejador de evento del control "Establecer fuente" (cambio de la propiedad OBJPROP_FONT).

```
virtual bool  OnSetFont()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

El manejador de evento del control "Establecer tamaño de la fuente" (cambio de la propiedad OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnCreate

El manejador de evento del control "Crear".

```
virtual bool  OnCreate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnShow

El manejador de evento del control "Mostrar".

```
virtual bool  OnShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnHide

El manejador de evento del control "Esconder".

```
virtual bool  OnHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnMove

El manejador de evento del control "Mover".

```
virtual bool  OnMove()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Klasse CBmpButton

Klasse CBmpButton ist eine Klasse des einfachen Steuerelements basierend auf das grafischen Objekt "Bitmap Label".

Beschreibung

Klasse CBmpButton wird für die Erstellung von Tasten mit einem grafischen Bild verwendet.

Deklaration

```
class CBmpButton : public CWndObj
```

Kopf

```
#include <Controls\BmpButton.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndObj
      CBmpButton

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

Erstellung	
Create	Erstellt ein Steuerelement

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Steuerelement-Parameter	
Border	Erhält/setzt die Eigenschaft "Border" eines Steuerelements
BmpNames	Setzt die Namen der BMP-Dateien zur Anzeige eines Steuerelements
BmpOffName	Erhält/Setzt den Namen der BMP-Datei zur Anzeige eines Steuerelements im OFF-Zustand
BmpOnName	Erhält/Setzt den Namen der BMP-Datei zur Anzeige eines Steuerelements im ON-Zustand
BmpPassiveName	Erhält/Setzt den Namen der BMP-Datei zur Anzeige eines Steuerelements im inaktiven Zustand
BmpActiveName	Erhält/Setzt den Namen der BMP-Datei zur Anzeige eines Steuerelements im aktiven Zustand
Eigenschaften	
Pressed	Erhält/setzt den Zustand eines Steuerelements
Locking	Erhält/setzt den Wert der Eigenschaft "Locking" eines Steuerelements
Behandlung von internen Ereignissen	
OnSetZOrder	Handler des internen Ereignisses "SetZOrder" eines Steuerelements
OnCreate	Handler des internen Ereignisses "Create" eines Steuerelements
OnShow	Handler des internen Ereignisses "Show" eines Steuerelements
OnHide	Handler des internen Ereignisses "Hide" eines Steuerelements
OnMove	Handler des internen Ereignisses "Move" eines Steuerelements
OnChange	Handler des internen Ereignisses "Change" eines Steuerelements
OnActivate	Handler des internen Ereignisses "Activate" eines Steuerelements
OnDeactivate	Handler des internen Ereignisses "Deactivate" eines Steuerelements
OnMouseDown	Handler des internen Ereignisses "MouseDown" eines Steuerelements
OnMouseUp	Handler des internen Ereignisses "MouseUp" eines Steuerelements

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Ein Beispiel für die Erstellung eines Panels mit einem Bitmap-Label

```
//+------------------------------------------------------------------+
//|                                          ControlsBmpButton.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CBmpButton"
#include <Controls\Dialog.mqh>
#include <Controls\BmpButton.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CBmpButton        m_bmpbutton1;                    // CBmpButton object
   CBmpButton        m_bmpbutton2;                    // CBmpButton object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateBmpButton1(void);
   bool              CreateBmpButton2(void);
   //--- handlers of the dependent controls events
   void              OnClickBmpButton1(void);
   void              OnClickBmpButton2(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_bmpbutton1,OnClickBmpButton1)
ON_EVENT(ON_CLICK,m_bmpbutton2,OnClickBmpButton2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateBmpButton1())
      return(false);
   if(!CreateBmpButton2())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "BmpButton1" button                                   |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_bmpbutton1.Create(m_chart_id,m_name+"BmpButton1",m_subwin,x1,y1,x2,y2))
      return(false);
//--- sets the name of bmp files of the control CBmpButton
   m_bmpbutton1.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "BmpButton2" fixed button                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_bmpbutton2.Create(m_chart_id,m_name+"BmpButton2",m_subwin,x1,y1,x2,y2))
      return(false);
//--- sets the name of bmp files of the control CBmpButton
   m_bmpbutton2.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton2))
      return(false);
   m_bmpbutton2.Locking(true);
//--- succeed
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton1(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton2(void)
  {
   if(m_bmpbutton2.Pressed())
      Comment(__FUNCTION__+" State of the control is: On");
   else
      Comment(__FUNCTION__+" State of the control is: Off");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CBmpButton.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Border (Método Get)

Obtiene la propiedad "Borde" (anchura del borde) del control.

```
int  Border()  const
```

Valor devuelto

La propiedad "Borde".

Border (Método Set)

Establece la propiedad "Borde" (anchura del borde) del control.

```
bool  Border(
   const int  value      // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo de la propiedad "Borde".

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

BmpNames

Establece el nombre de los archivos bmp del control

```
bool  BmpNames(
   const string  off="",      // nombre del archivo
   const string  on=""        // nombre del archivo
   )
```

Parámetros

off=""

[in] Nombre del archivo bmp para el estado OFF.

on=""

[in] Nombre del archivo bmp para el estado ON.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

BmpOffName (Método Get)

Obtiene el nombre del archivo bmp para el estado OFF.

```
string  BmpOffName()  const
```

Valor devuelto

Nombre del archivo bmp para el estado OFF.

BmpOffName (Método Set)

Establece el nombre del archivo bmp para el estado OFF.

```
bool  BmpOffName(
   const string  name      // nombre del archivo
   )
```

Parámetros

name

[in] Nombre del archivo bmp para el estado OFF.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

BmpOnName (Método Get)

Obtiene el nombre del archivo bmp para el estado ON.

```
string  BmpOnName()  const
```

Valor devuelto

Nombre del archivo bmp para el estado ON.

BmpOnName (Método Set)

Establece el nombre del archivo bmp para el estado ON.

```
bool  BmpOnName(
   const string  name      // nombre del archivo
   )
```

Parámetros

name

[in] Nombre del archivo bmp para el estado ON.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

BmpPassiveName (Método Get)

Obtiene el nombre del archivo bmp para el estado pasivo de control.

```
string  BmpPassiveName()  const
```

Valor devuelto

Nombre del archivo bmp para el estado pasivo de control.

BmpPassiveName (Método Set)

Establece el nombre del archivo bmp para el estado pasivo.

```
bool  BmpPassiveName(
   const string  name      // nombre del archivo
   )
```

Parámetros

name

[in] Nombre del archivo bmp para el estado pasivo de control.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

BmpActiveName (Método Get)

Obtiene el nombre del archivo bmp para el estado activo.

```
string  BmpActiveName()  const
```

Valor devuelto

Nombre del archivo bmp para el estado activo.

Nota

El control se activa al pasar por encima el cursor del ratón.

BmpActiveName (Método Set)

Establece el nombre del archivo bmp para el estado activo.

```
bool  BmpActiveName(
   const string  name      // nombre del archivo
   )
```

Parámetros

name

[in] Nombre del archivo bmp para el estado activo.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Pressed (Método Get)

Obtiene el estado del control (propiedad "Presionado").

```
bool  Pressed()  const
```

Valor devuelto

Estado del control.

Pressed (Método Set)

Establece el estado del control (propiedad "Presionado").

```
bool  Pressed(
   const bool  pressed      // estado nuevo
   )
```

Parámetros

pressed

[in] Nuevo estado del control.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Locking (Método Get)

Obtiene la propiedad "Bloqueo" del control.

```
bool  Locking()  const
```

Valor devuelto

El valor de la propiedad "Bloqueo".

Locking (Método Set)

Establece el nuevo valor de la propiedad "Bloqueo" del control.

```
void  Locking(
   const bool  locking      // valor nuevo
   )
```

Parámetros

locking

[in] Nuevo valor de la propiedad "Bloqueo".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

El manejador de evento del control "Establecer orden Z" (cambio de la propiedad OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnCreate

El manejador de evento del control "Crear".

```
virtual bool  OnCreate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnShow

El manejador de evento del control "Mostrar".

```
virtual bool  OnShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnHide

El manejador de evento del control "Esconder".

```
virtual bool  OnHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnMove

El manejador de evento del control "Mover".

```
virtual bool  OnMove()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChange

El manejador de evento del control "Cambiar".

```
virtual bool  OnChange()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnActivate

El manejador de evento del control "Activar".

```
virtual bool  OnActivate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnDeactivate

El manejador de evento del control "Desactivar".

```
virtual bool  OnDeactivate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnMouseDown

El manejador de evento del control "Ratón abajo".

```
virtual bool  OnMouseDown()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Ratón abajo" ocurre al hacer clic con el botón izquierdo del ratón sobre el control.

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

El manejador de evento del control "Ratón arriba" (se suelta el botón izquierdo del ratón).

```
virtual bool  OnMouseUp()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Ratón arriba" ocurre cuando se suelta el botón izquierdo del ratón sobre el control.

© 2000-2025, MetaQuotes Ltd.

Klasse CButton

Klasse CButton ist eine Klasse des einfachen Steuerelements basierend auf das grafischen Objekt "Taste".

Beschreibung

Klasse CButton wird für die Erstellung von einfachen Tasten verwendet.

Deklaration

```
class CButton : public CWndObj
```

Kopf

```
#include <Controls\Button.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndObj
      CButton

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

Erstellung	
Create	Erstellt ein Steuerelement

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Eigenschaften	
Pressed	Erhält/Setzt den Wert der Eigenschaft "Pressed".
Locking	Erhält/Setzt den Wert der Eigenschaft "Locking".
Parameteränderungenbehandlung	
OnSetText	Handler des Ereignisses "SetText" eines Steuerelements
OnSetColor	Handler des Ereignisses "SetColor" eines Steuerelements
OnSetColorBackground	Handler des Ereignisses "SetColorBackground" eines Steuerelements
OnSetColorBorder	Handler des Ereignisses "SetColorBorder" eines Steuerelements
OnSetFont	Handler des Ereignisses "SetFont" eines Steuerelements
OnSetFontSize	Handler des Ereignisses "SetFontSize" eines Steuerelements
Behandlung von internen Ereignissen	
OnCreate	Handler des internen Ereignisses "Create" eines Steuerelements
OnShow	Handler des internen Ereignisses "Show" eines Steuerelements
OnHide	Handler des internen Ereignisses "Hide" eines Steuerelements
OnMove	Handler des internen Ereignisses "Move" eines Steuerelements
OnResize	Handler des internen Ereignisses "Resize" eines Steuerelements
OnMouseDown	Handler des internen Ereignisses "MouseDown" eines Steuerelements
OnOnMouseUp	Handler des internen Ereignisses "MouseUp" eines Steuerelements

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndObj

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Ein Beispiel für die Erstellung eines Panels mit einem Button:

```
//+------------------------------------------------------------------+
//|                                                ControlsButton.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Das Panel der Anzeige und der Dialoge. Demonstration der CButto
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CButton           m_button1;                     // the button object
   CButton           m_button2;                     // the button object
   CButton           m_button3;                     // the fixed button object
```

© 2000-2025, MetaQuotes Ltd.

```
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateButton1(void);
   bool              CreateButton2(void);
   bool              CreateButton3(void);
   //--- handlers of the dependent controls events
   void              OnClickButton1(void);
   void              OnClickButton2(void);
   void              OnClickButton3(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_button1,OnClickButton1)
ON_EVENT(ON_CLICK,m_button2,OnClickButton2)
ON_EVENT(ON_CLICK,m_button3,OnClickButton3)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateButton1())
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!CreateButton2())
      return(false);
   if(!CreateButton3())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button1" button                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_button1.Create(m_chart_id,m_name+"Button1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button1.Text("Button1"))
      return(false);
   if(!Add(m_button1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button2" button                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_button2.Create(m_chart_id,m_name+"Button2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button2.Text("Button2"))
      return(false);
   if(!Add(m_button2))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button3" fixed button                                |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton3(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_button3.Create(m_chart_id,m_name+"Button3",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button3.Text("Locked"))
      return(false);
   if(!Add(m_button3))
      return(false);
   m_button3.Locking(true);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton1(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton2(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton3(void)
  {
   if(m_button3.Pressed())
      Comment(__FUNCTION__+" Status des Elements: On");
   else
      Comment(__FUNCTION__+" Status des Elements: Off");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Kommentare löschen
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea un nuevo control CButton.

```
virtual bool  Create(
   const long     chart,      // Identificador gráfico
   const string   name,       // nombre
   const int      subwin,     // subventana gráfica
   const int      x1,         // coordenada x1
   const int      y1,         // coordenada y1
   const int      x2,         // coordenada x2
   const int      y2          // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Pressed (Método Get)

Obtiene el estado del control (propiedad "Presionado").

```
bool  Pressed()  const
```

Valor devuelto

Estado del control.

Pressed (Método Set)

Establece el estado del control (propiedad "Presionado").

```
bool  Pressed(
   const bool  pressed      // estado nuevo
   )
```

Parámetros

pressed

[in] Nuevo estado del control.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Locking (Método Get)

Obtiene la propiedad "Bloqueo" del control.

```
bool  Locking()  const
```

Valor devuelto

El valor de la propiedad "Bloqueo".

Locking (Método Set)

Establece la nueva propiedad "Bloqueo" del control.

```
void  Locking(
   const bool  locking      // valor nuevo
   )
```

Parámetros

locking

[in] Nuevo valor de la propiedad "Bloqueo".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

OnSetText

El manejador de evento del control "Establecer texto" (cambio de la propiedad OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

El manejador de evento del control "Establecer color" (cambio de la propiedad OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

El manejador de evento del control "Establecer color de fondo" (cambio de la propiedad OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBorder

El manejador de evento del control "Establecer color del borde" (cambio de la propiedad OBJPROP_BORDER_COLOR).

```
virtual bool  OnSetColorBackground()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

El manejador de evento del control "Establecer fuente" (cambio de la propiedad OBJPROP_FONT).

```
virtual bool  OnSetFont()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

El manejador de evento del control "Establecer tamaño de la fuente" (cambio de la propiedad OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnCreate

El manejador de evento del control "Crear".

```
virtual bool  OnCreate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnShow

El manejador de evento del control "Mostrar".

```
virtual bool  OnShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnHide

El manejador de evento del control "Esconder".

```
virtual bool  OnHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnMove

El manejador de evento del control "Mover".

```
virtual bool  OnMove()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnResize

El manejador de evento del control "Cambiar tamaño".

```
virtual bool  OnResize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnMouseDown

El manejador de evento del control "Ratón abajo".

```
virtual bool  OnMouseDown()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Ratón abajo" ocurre al hacer clic con el botón izquierdo del ratón sobre el control.

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

El manejador de evento del control "Ratón arriba" (se suelta el botón izquierdo del ratón).

```
virtual bool  OnMouseUp()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Ratón arriba" ocurre cuando se suelta el botón izquierdo del ratón sobre el control.

© 2000-2025, MetaQuotes Ltd.

Klasse CEdit

Klasse CEdit ist eine Klasse des einfachen Steuerelements basierend auf das grafischen Objekt "Eingabefeld".

Beschreibung

Klasse CEdit wird für die Erstellung von bearbeitbaren Textfeldern verwendet.

Deklaration

```
class CEdit : public CWndObj
```

Kopf

```
#include <Controls\Edit.mqh>
```

Jerarquía de herencia

CObject
CWnd
CWndObj
CEdit

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

Erstellung	
Create	Erstellt ein Steuerelement

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Eigenschaften	
ReadOnly	Erhält/setzt den Wert der Eigenschaft "ReadOnly" eines Steuerelements
TextAlign	Erhält/setzt den Wert der Eigenschaft "TextAlign" eines Steuerelements
Behandlung von Objekt-Ereignissen	
OnObjectEndEdit	Virtueller Handler des Ereignisses CHARTEVENT_OBJECT_ENDEDIT des Chartobjekts
Parameteränderungenbehandlung	
OnSetText	Handler des Ereignisses "SetText" eines Steuerelements
OnSetColor	Handler des Ereignisses "SetColor" eines Steuerelements
OnSetColorBackground	Handler des Ereignisses "SetColorBackground" eines Steuerelements
OnSetColorBorder	Handler des Ereignisses "SetColorBorder" eines Steuerelements
OnSetFont	Handler des Ereignisses "SetFont" eines Steuerelements
OnSetFontSize	Handler des Ereignisses "SetFontSize" eines Steuerelements
OnSetZOrder	Handler des Ereignisses "SetZOrder" eines Steuerelements
Behandlung von internen Ereignissen	
OnCreate	Handler des internen Ereignisses "Create" eines Steuerelements
OnShow	Handler des internen Ereignisses "Show" eines Steuerelements
OnHide	Handler des internen Ereignisses "Hide" eines Steuerelements
OnMove	Handler des internen Ereignisses "Move" eines Steuerelements
OnResize	Handler des internen Ereignisses "Resize" eines Steuerelements
OnChange	Handler des internen Ereignisses "Change" eines Steuerelements
OnClick	Handler des internen Ereignisses "Click" eines Steuerelements

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndObj

Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Ein Beispiel für die Erstellung eines Panels mit einem editierbaren Textfeld:

```
//+------------------------------------------------------------------+
//|                                                ControlsEdit.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Das Panel der Anzeige und der Dialoge. Demonstration der CEdit
#include <Controls\Dialog.mqh>
#include <Controls\Edit.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
```

© 2000-2025, MetaQuotes Ltd.

```
  {
private:
   CEdit             m_edit;                          // CEdit Objekt

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler

protected:
   //--- create dependent controls
   bool              CreateEdit(void);
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateEdit())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the display field                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateEdit(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=ClientAreaWidth()-INDENT_RIGHT;
```

© 2000-2025, MetaQuotes Ltd.

```
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_edit.Create(m_chart_id,m_name+"Edit",m_subwin,x1,y1,x2,y2))
      return(false);
//--- erlauben, den Inhalt zu bearbeiten
   if(!m_edit.ReadOnly(false))
      return(false);
   if(!Add(m_edit))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Kommentare löschen
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CEdit.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ReadOnly (Método Get)

Obtiene la propiedad "Solo lectura" del control.

```
bool  ReadOnly()
```

Valor devuelto

El valor de la propiedad "Solo lectura".

ReadOnly (Método Set)

Establece el valor de la propiedad "Solo lectura" del control.

```
bool  ReadOnly(
   const bool  flag      // valores nuevos
   )
```

Parámetros

flag

[in] Valor nuevo de la propiedad "Solo lectura".

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

TextAlign (Método Get)

Obtiene el valor de la propiedad "Alinear texto" (modo de alineación del texto) del control.

```
ENUM_ALIGN_MODE  TextAlign() const
```

Valor devuelto

Valor de la propiedad "Alinear texto" del control.

TextAlign (Método Set)

Establece el valor nuevo de la propiedad "Alinear texto" (modo de alineación del texto) del control.

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // valor nuevo
   )
```

Parámetros

align

[in] Valor nuevo de la propiedad "Alinear texto".

Valor devuelto

true si se ejecuta correctamente, false si la propiedad no ha cambiado.

© 2000-2025, MetaQuotes Ltd.

OnObjectEndEdit

El manejador de evento CHARTEVENT_OBJECT_ENDEDIT.

```
virtual bool  OnObjectEndEdit()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetText

El manejador de evento del control "Establecer texto" (cambio de la propiedad OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

El manejador de evento del control "Establecer color" (cambio de la propiedad OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

El manejador de evento del control "Establecer color de fondo" (cambio de la propiedad OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBorder

El manejador de evento del control "Establecer color de borde" (cambio de la propiedad OBJPROP_BORDER_COLOR).

```
virtual bool  OnSetColorBackground()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

El manejador de evento del control "Establecer fuente" (cambio de la propiedad OBJPROP_FONT).

```
virtual bool  OnSetFont()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

El manejador de evento del control "Establecer tamaño de la fuente" (cambio de la propiedad OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

El manejador de evento del control "Establecer orden Z" (cambio de la propiedad OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnCreate

El manejador de evento del control "Crear".

```
virtual bool  OnCreate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnShow

El manejador de evento del control "Mostrar".

```
virtual bool  OnShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnHide

El manejador de evento del control "Esconder".

```
virtual bool  OnHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnMove

El manejador de evento del control "Mover".

```
virtual bool  OnMove()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnResize

El manejador de evento del control "Cambiar tamaño".

```
virtual bool  OnResize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChange

El manejador de evento del control "Cambiar".

```
virtual bool  OnChange()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClick

El manejador de evento del control "Clicar" (clic con el botón izquierdo del ratón).

```
virtual bool  OnClick()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Klasse CPanel

Klasse CPanel ist eine Klasse des einfachen Steuerelements basierend auf das grafischen Objekt "Rechteck-Label".

Beschreibung

Klasse CPanel wird für die visuelle Gruppierung von funktionell verwandten homogenen Elementen verwendet.

Deklaration

```
class CPanel : public CWndObj
```

Kopf

```
#include <Controls\Panel.mqh>
```

Jerarquía de herencia

CObject
CWnd
CWndObj
CPanel

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Create	Erstellt ein Steuerelement
Parameter des Chartobjekts	
BorderType	Erhält den Wert des Parameters "Border" (Rahmentyp) des Chartobjekts
Behandlung von Objekt-Ereignissen	
OnSetText	Handler des Ereignisses "SetText" eines Steuerelements
OnSetColorBackground	Handler des Ereignisses "SetColorBackground" eines Steuerelements
OnSetColorBorder	Handler des Ereignisses "SetColorBorder" eines Steuerelements
Behandlung von internen Ereignissen	
OnCreate	Handler des internen Ereignisses "Create" eines Steuerelements
OnShow	Handler des internen Ereignisses "Show" eines Steuerelements
OnHide	Handler des internen Ereignisses "Hide" eines Steuerelements
OnMove	Handler des internen Ereignisses "Move" eines Steuerelements
OnResize	Handler des internen Ereignisses "Resize" eines Steuerelements
OnChange	Handler des internen Ereignisses "Change" eines Steuerelements

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Ein Beispiel für die Erstellung eines Panels mit einem Rechteck-Label:

```
//+------------------------------------------------------------------+
//|                                                ControlsPanel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Das Panel der Anzeige und der Dialoge. Demonstration der CPanel
#include <Controls\Dialog.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const

protected:
   //--- create dependent controls
   bool              CreatePanel(void);
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreatePanel())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CPanel"                                              |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePanel(void)
  {
//--- coordinates
   int x1=20;
   int y1=20;
   int x2=ExtDialog.Width()/3;
   int y2=ExtDialog.Height()/3;
//--- create
   if(!my_white_border.Create(0,ExtDialog.Name()+"MyWhiteBorder",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!my_white_border.ColorBackground(CONTROLS_DIALOG_COLOR_BG))
      return(false);
   if(!my_white_border.ColorBorder(CONTROLS_DIALOG_COLOR_BORDER_LIGHT))
      return(false);
   if(!ExtDialog.Add(my_white_border))
      return(false);
   my_white_border.Alignment(WND_ALIGN_CLIENT,0,0,0,0);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//---
CPanel   my_white_border;        // object CPanel
bool     pause=true;             // true - Pause
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   EventSetTimer(3);
   pause=true;
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Kommentare löschen
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
//+------------------------------------------------------------------+
//|  Timer                                                           |
//+------------------------------------------------------------------+
void OnTimer()
  {
   pause=!pause;
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CPanel.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

BorderType (Método Get)

Obtiene la propiedad "Tipo de borde" del objeto gráfico.

```
ENUM_BORDER_TYPE  BorderType()
```

Valor devuelto

El valor de la propiedad "Tipo de borde".

BorderType (Método Set)

Establece el nuevo valor de la propiedad "Tipo de borde" del objeto gráfico.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // valor
   )
```

Parámetros

type

[in] Valor nuevo de la propiedad "Tipo de borde".

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetText

El manejador de evento del control "Establecer texto" (cambio de la propiedad OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

El manejador de evento del control "Establecer color de fondo" (cambio de la propiedad OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBorder

El manejador de evento del control "Establecer color de borde" (cambio de la propiedad OBJPROP_BORDER_COLOR).

```
virtual bool  OnSetColorBackground()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnCreate

El manejador de evento del control "Crear".

```
virtual bool  OnCreate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnShow

El manejador de evento del control "Mostrar".

```
virtual bool  OnShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnHide

El manejador de evento del control "Esconder".

```
virtual bool  OnHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnMove

El manejador de evento del control "Mover".

```
virtual bool  OnMove()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnResize

El manejador de evento del control "Cambiar tamaño".

```
virtual bool  OnResize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChange

El manejador de evento del control "Cambiar".

```
virtual bool  OnChange()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Klasse CPicture

Klasse CPicture ist eine Klasse des einfachen Steuerelements basierend auf das grafischen Objekt "Bitmap Label".

Beschreibung

Klasse CPicture wird für die Erstellung von einfachen Grafiken verwendet.

Deklaration

```
class CPicture : public CWndObj
```

Kopf

```
#include <Controls\Picture.mqh>
```

Jerarquía de herencia

CObject
CWnd
CWndObj
CPicture

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

Erstellung	
Create	Erstellt ein Steuerelement

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Eigenschaften des Chartobjekts	
Border	Erhält/setzt die Rahmenbreite eines Chartobjekts
BmpName	Erhält/setzt den Namen der BMP-Dateien zur Anzeige eines Steuerelements
Behandlung von internen Ereignissen	
OnCreate	Handler des internen Ereignisses "Create" eines Steuerelements
OnShow	Handler des internen Ereignisses "Show" eines Steuerelements
OnHide	Handler des internen Ereignisses "Hide" eines Steuerelements
OnMove	Handler des internen Ereignisses "Move" eines Steuerelements
OnChange	Handler des internen Ereignisses "Change" eines Steuerelements

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos heredados de la clase CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Ein Beispiel für die Erstellung eines Panels mit einem Bitmap-Label:

```
//+------------------------------------------------------------------+
//|                                                ControlsPicture.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Das Panel der Anzeige und der Dialoge. Demonstration der CPictu
#include <Controls\Dialog.mqh>
#include <Controls\Picture.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CPicture          m_picture;                      // CPicture object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreatePicture(void);
   //--- handlers of the dependent controls events
   void              OnClickPicture(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_picture,OnClickPicture)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreatePicture())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Picture"                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePicture(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+32;
   int y2=y1+32;
//--- create
   if(!m_picture.Create(m_chart_id,m_name+"Picture",m_subwin,x1,y1,x2,y2))
      return(false);
//--- benennen wir die bmp-Dateien für die Anzeige des CPicture Steuerelements
   m_picture.BmpName("\\Images\\euro.bmp");

   if(!Add(m_picture))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickPicture(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Kommentare löschen
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CPicture.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Border (Método Get)

Obtiene la propiedad "Borde" (anchura del borde) del control.

```
int  Border()  const
```

Valor devuelto

La propiedad "Borde".

Border (Método Set)

Establece la propiedad "Borde" (anchura del borde) del control.

```
bool  Border(
   const int  value      // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo de la propiedad "Borde".

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

BmpName (Método Get)

Obtiene el nombre del archivo bmp del control.

```
string  BmpName()  const
```

Valor devuelto

Nombre del archivo bmp del control.

BmpName (Método Set)

Establece el nombre del archivo bmp del control.

```
bool  BmpName(
   const string  name      // nombre del archivo
   )
```

Parámetros

name

[in] Nombre del archivo bmp del control.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnCreate

El manejador de evento del control "Crear".

```
virtual bool  OnCreate()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnShow

El manejador de evento del control "Mostrar".

```
virtual bool  OnShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnHide

El manejador de evento del control "Esconder".

```
virtual bool  OnHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnMove

El manejador de evento del control "Mover".

```
virtual bool  OnMove()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChange

El manejador de evento del control "Cambiar".

```
virtual bool  OnChange()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CScroll

CScroll es una clase base para la creación de barras de desplazamiento.

Descripción

CScroll es un control complejo (con controles dependientes) que contiene la funcionalidad base para crear barras de desplazamiento. La clase base no se utiliza como un control separado en sí misma, sino que dos de sus herederos (las clases CScrollV y CScrollH) se utilizan como controles.

Declaración

```
class CScroll : public CWndContainer
```

Título

```
#include <Controls\Scrolls.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndContainer
      CScroll

Descendientes directos

CScrollH, CScrollV

Métodos de la clase

Creación	
Create	Crea el control
Manejadores de evento del objeto gráfico	
OnEvent	Manejador de todos los eventos gráficos
Propiedades	
MinPos	Obtiene/Establece la posición mínima
MaxPos	Obtiene/Establece la posición máxima
CurrPos	Obtiene/Establece la posición actual
Creación de controles dependientes	
CreateBack	Crea el botón de fondo
CreateInc	Crea el botón de incremento de la barra de desplazamiento

© 2000-2025, MetaQuotes Ltd.

Creación	
CreateDec	Crea el botón de decremento de la barra de desplazamiento
CreateThumb	Crea el botón (se puede arrastrar) de la barra de desplazamiento
Manejadores de eventos de controles dependientes	
OnClickInc	Manejador de evento utilizado para el incremento de los eventos de botón
OnClickDec	Manejador de evento utilizado para el decremento de los eventos de botón
Manejadores de evento internos	
OnShow	Manejador de evento "Crear"
OnHide	Manejador de evento "Esconder"
OnChangePos	Manejador de evento "Cambiar posición"
Controladores de arrastre del objeto	
OnThumbDragStart	Manejador de evento "Inicio de arrastre del pulgar"
OnThumbDragProcess	Manejador de evento "Proceso de arrastre del pulgar"
OnThumbDragEnd	Manejador de evento "Fin de arrastre del pulgar"
Posición	
CalcPos	Obtiene la posición de la barra de desplazamiento por medio de la coordenada

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CScroll.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

MinPos (Método Get)

Obtiene el valor "MinPos" (posición mínima) del control CScroll.

```
int  MinPos()  const
```

Valor devuelto

Nuevo valor de la propiedad "MinPos".

MinPos (Método Set)

Establece el valor de la propiedad "MinPos" (posición mínima) del control CScroll.

```
void  MinPos(
   const int  value      // valor nuevo
   )
```

Parámetros

value

[in] Nuevo valor de la propiedad "MinPos".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

MaxPos (Método Get)

Obtiene el valor "MaxPos" (posición máxima) del control CScroll.

```
int  MaxPos()  const
```

Valor devuelto

Nuevo valor de la propiedad "MaxPos".

MaxPos (Método Set)

Establece el valor "MaxPos" (posición máxima) del control CScroll.

```
void  MaxPos(
   const int  value      // valor nuevo
   )
```

Parámetros

value

[in] Nuevo valor de la propiedad "MaxPos".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

CurrPos (Método Get)

Obtiene el valor de la posición actual del control CScroll.

```
int  CurrPos()  const
```

Valor devuelto

Nuevo valor de la propiedad "Posición actual".

CurrPos (Método Set)

Establece el valor de la posición actual del control CScroll.

```
void  CurrPos(
   const int  value      // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo de la propiedad posición actual.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

CreateBack

Crea el botón de fondo del control CScroll.

```
virtual bool  CreateBack()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateInc

Crea el botón de incremento del control CScroll.

```
virtual bool  CreateInc()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateDec

Crea el botón de decremento del control CScroll.

```
virtual bool  CreateDec()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateThumb

Crea el botón pulgar (se puede arrastrar) del control CScroll.

```
virtual bool  CreateThumb()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickInc

El manejador de evento del control "ClickInc" (clic con el botón izquierdo del ratón sobre el botón de incremento).

```
virtual bool  OnClickInc()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickDec

El manejador de evento del control "ClickDec" (clic con el botón izquierdo del ratón sobre el botón de decremento).

```
virtual bool  OnClickDec()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnShow

El manejador de evento del control "Mostrar".

```
virtual bool  OnShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnHide

El manejador de evento del control "Esconder".

```
virtual bool  OnHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChangePos

El manejador de evento del control "Cambiar posición".

```
virtual bool  OnChangePos()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragStart

El manejador de evento del control "ThumbDragStart" (empezar desplazamiento).

```
virtual bool  OnThumbDragStart()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "ThumbDragStart" se produce al comienzo de la operación de arrastrar.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragProcess

El manejador de evento del control "ThumbDragProcess".

```
virtual bool  OnThumbDragProcess(
   const int  x,      // coordenada x
   const int  y       // coordenada y
   )
```

Parámetros

x

[in] Coordenada X actual del puntero del ratón.

y

[in] Coordenada Y actual del puntero del ratón.

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "ThumbDragProcess" se produce cuando se mueve el control de la barra de desplazamiento.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragEnd

El manejador de evento del control "ThumbDragEnd" (proceso de arrastre finalizado).

```
virtual bool  OnThumbDragEnd()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "ThumbDragEnd" se produce cuando finaliza la operación de arrastrar la barra de desplazamiento.

© 2000-2025, MetaQuotes Ltd.

CalcPos

Obtiene la posición de la barra de desplazamiento por medio de la coordenada.

```
virtual int  CalcPos(
   const int      coord      // coordenada
   )
```

Parámetros

coord

[in] Coordenada de la barra de desplazamiento.

Valor devuelto

Posición de la barra de desplazamiento.

© 2000-2025, MetaQuotes Ltd.

Klasse CScrollV

CScrollV ist eine Klasse des kombinierten Steuerelements "Senkrechte Scrollleiste".

Beschreibung

Klasse CScrollV wird für die Erstellung von senkrechten Scrollleisten verwendet.

Deklaration

```
class CScrollV : public CScroll
```

Kopf

```
#include <Controls\Scrolls.mqh>
```

Jerarquía de herencia

CObject
CWnd
CWndContainer
CScroll
CScrollV

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

© 2000-2025, MetaQuotes Ltd.

Erstellung von unterlegenden Steuerelementen	
CreateInc	Erstellt eine Taste, die die Position der Bildlaufleiste erhöht
CreateDec	Erstellt eine Taste, die die Position der Bildlaufleiste reduziert
CreateThumb	Erstellt einen Schieberegler der aktuellen Position der Bildlaufleiste
Behandlung von internen Ereignissen	
OnResize	Handler des internen Ereignisses "Resize" eines Steuerelements
OnChangePos	Handler des internen Ereignisses "ChangePosition" eines Steuerelements
Ereignisse von Objekt-Bewegung	
OnThumbDragStart	Handler des Ereignisses "ThumbDragStart" eines Steuerelements
OnThumbDragProcess	Handler des Ereignisses "ThumbDragProcess" eines Steuerelements
OnThumbDragEnd	Handler des Ereignisses "ThumbDragEnd" eines Steuerelements
Die Berechnung der Position nach den Koordinaten	
CalcPos	Erhält die Position der Bildlaufleiste nach den Koordinaten

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Métodos heredados de la clase CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Ein Beispiel für die Erstellung eines Panels mit der waagerechten Scroll-Funktion:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                ControlsScrollV.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Das Panel der Anzeige und der Dialoge. Demonstration der CScrol
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollV          m_scroll_v;                     // CScrollV Objekt

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
```

```
   bool              CreateScrollV(void);
   //--- handlers of the dependent controls events
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateScrollV())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the CScrollsV object                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollV(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+18;
   int y2=y1+LIST_HEIGHT;
//--- create
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollV",m_subwin,x1,y1,x2,y2))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- set up the scrollbar
   m_scroll_v.MinPos(0);
//--- set up the scrollbar
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
      return(false);
   Comment("Position der Scrollleiste ",m_scroll_v.CurrPos());
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position der Scrollleiste ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position der Scrollleiste ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Kommentare löschen
   Comment("");
```

© 2000-2025, MetaQuotes Ltd.

```
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateInc

Crea el botón de incremento del control.

```
virtual bool  CreateInc()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateDec

Crea el botón de decremento del control.

```
virtual bool  CreateDec()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateThumb

Crea el botón pulgar (se puede arrastrar) del control.

```
virtual bool  CreateThumb()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnResize

El manejador de evento del control "Cambiar tamaño".

```
virtual bool  OnResize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChangePos

El manejador de evento del control "Cambiar posición".

```
virtual bool  OnChangePos()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragStart

El manejador de evento del control "ThumbDragStart" (empezar desplazamiento).

```
virtual bool  OnThumbDragStart()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "ThumbDragStart" se produce al comienzo de la operación de arrastrar.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragProcess

El manejador de evento del control "ThumbDragProcess".

```
virtual bool  OnThumbDragProcess(
   const int  x,      // coordenada x
   const int  y       // coordenada y
   )
```

Parámetros

x

[in] Coordenada X actual del puntero del ratón.

y

[in] Coordenada Y actual del puntero del ratón.

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "ThumbDragProcess" se produce cuando se mueve el control de la barra de desplazamiento.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragEnd

El manejador de evento del control "ThumbDragEnd" (proceso de arrastre finalizado).

```
virtual bool  OnThumbDragEnd()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "ThumbDragEnd" se produce cuando finaliza la operación de arrastrar la barra de desplazamiento.

© 2000-2025, MetaQuotes Ltd.

CalcPos

Obtiene la posición de la barra de desplazamiento por medio de la coordenada.

```
virtual int  CalcPos(
   const int      coord      // coordenada
   )
```

Parámetros

coord

[in] Coordenada de la barra de desplazamiento.

Valor devuelto

Posición de la barra de desplazamiento.

© 2000-2025, MetaQuotes Ltd.

Klasse CScrollH

CScrollH ist eine Klasse des kombinierten Steuerelements "Waagerechte Scrollleiste".

Beschreibung

Klasse CScrollH wird für die Erstellung von waagerechten Scrollleisten verwendet.

Deklaration

```
class CScrollH : public CScroll
```

Kopf

```
#include <Controls\Scrolls.mqh>
```

Jerarquía de herencia

CObject
CWnd
CWndContainer
CScroll
CScrollH

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

© 2000-2025, MetaQuotes Ltd.

Erstellung von unterlegenden Steuerelementen	
CreateInc	Erstellt eine Taste, die die Position der Bildlaufleiste erhöht
CreateDec	Erstellt eine Taste, die die Position der Bildlaufleiste reduziert
CreateThumb	Erstellt einen Schieberegler der aktuellen Position der Bildlaufleiste
Behandlung von internen Ereignissen	
OnResize	Handler des internen Ereignisses "Resize" eines Steuerelements
OnChangePos	Handler des internen Ereignisses "ChangePosition" eines Steuerelements
Ereignisse von Objekt-Bewegung	
OnThumbDragStart	Handler des Ereignisses "ThumbDragStart" eines Steuerelements
OnThumbDragProcess	Handler des Ereignisses "ThumbDragProcess" eines Steuerelements
OnThumbDragEnd	Handler des Ereignisses "ThumbDragEnd" eines Steuerelements
Die Berechnung der Position nach den Koordinaten	
CalcPos	Erhält die Position der Bildlaufleiste nach den Koordinaten

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Métodos heredados de la clase CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Ein Beispiel für die Erstellung eines Panels mit der horizontalen Scroll-Funktion:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                 ControlsScrollH.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Das Panel der Anzeige und der Dialoge. Die Demonstration der CS
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollH          m_scroll_v;                     // CScrollH Objekt

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
```

```
   bool              CreateScrollsH(void);
   //--- handlers of the dependent controls events
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateScrollsH())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the CScrollsH object                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollsH(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+3*BUTTON_WIDTH;
   int y2=y1+18;
//--- create
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollsH",m_subwin,x1,y1,x2,y2))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- set up the scrollbar
   m_scroll_v.MinPos(0);
//--- set up the scrollbar
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
      return(false);
   Comment("Position der Scrollleiste ",m_scroll_v.CurrPos());
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position der Scrollleiste ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position der Scrollleiste ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Kommentare löschen
   Comment("");
```

© 2000-2025, MetaQuotes Ltd.

```
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateInc

Crea el botón de incremento del control.

```
virtual bool  CreateInc()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateDec

Crea el botón de decremento del control.

```
virtual bool  CreateDec()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateThumb

Crea el botón pulgar (se puede arrastrar) del control.

```
virtual bool  CreateThumb()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnResize

El manejador de evento del control "Cambiar tamaño".

```
virtual bool  OnResize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChangePos

El manejador de evento del control "Cambiar posición".

```
virtual bool  OnChangePos()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragStart

El manejador de evento del control "ThumbDragStart" (empezar desplazamiento).

```
virtual bool  OnThumbDragStart()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "ThumbDragStart" se produce al comienzo de la operación de arrastrar.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragProcess

El manejador de evento del control "ThumbDragProcess".

```
virtual bool  OnThumbDragProcess(
   const int  x,      // coordenada x
   const int  y       // coordenada y
   )
```

Parámetros

x

[in] Coordenada X actual del puntero del ratón.

y

[in] Coordenada Y actual del puntero del ratón.

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "ThumbDragProcess" se produce cuando se mueve el control de la barra de desplazamiento.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragEnd

El manejador de evento del control "ThumbDragEnd" (proceso de arrastre finalizado).

```
virtual bool  OnThumbDragEnd()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "ThumbDragEnd" se produce cuando finaliza la operación de arrastrar la barra de desplazamiento.

© 2000-2025, MetaQuotes Ltd.

CalcPos

Obtiene la posición de la barra de desplazamiento por medio de la coordenada.

```
virtual int  CalcPos(
   const int      coord      // coordenada
   )
```

Parámetros

coord

[in] Coordenada de la barra de desplazamiento.

Valor devuelto

Posición de la barra de desplazamiento.

© 2000-2025, MetaQuotes Ltd.

CWndClient

CWndClient es una clase del control complejo "Área cliente" (con controles dependientes). Es la clase base para la creación de barras de desplazamiento.

Descripción

CWndClient implementa la funcionalidad para crear áreas cliente con barras de desplazamiento.

Declaración

```
class CWndClient : public CWndContainer
```

Título

```
#include <Controls\WndClient.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndContainer
      CWndClient

Descendientes directos

CCheckGroup, CListView, CRadioGroup

Métodos de la clase

Creación	
Create	Crea el control
Manejador de evento gráfico	
OnEvent	Manejador de todos los eventos gráficos
Propiedades	
ColorBackground	Establece el color de fondo
ColorBorder	Establece el color del borde
BorderType	Establece el tipo de borde
Configuraciones	
VScrolled	Obtiene/Establece la bandera que indica el uso de la barra de desplazamiento vertical
HScrolled	Obtiene/Establece la bandera que indica el uso de la barra de desplazamiento horizontal
Controles dependientes	

© 2000-2025, MetaQuotes Ltd.

Creación	
CreateBack	Crea el fondo de la barra de desplazamiento
CreateScrollV	Crea la barra de desplazamiento vertical
CreateScrollH	Crea la barra de desplazamiento horizontal
Manejadores de evento internos	
OnResize	Manejador de evento "Cambiar tamaño"
Manejadores de eventos de controles dependientes	
OnVScrollShow	Manejador de evento "Mostrar" (virtual) del control dependiente VScroll
OnVScrollHide	Manejador de evento "Esconder" (virtual) del control dependiente VScroll
OnHScrollShow	Manejador de evento "Mostrar" (virtual) del control dependiente HScroll
OnHScrollHide	Manejador de evento "Esconder" (virtual) del control dependiente HScroll
OnScrollLineDown	Manejador de evento "Desplazar línea hacia abajo" (virtual) del control dependiente VScroll
OnScrollLineUp	Manejador de evento "Desplazar línea hacia arriba" (virtual) del control dependiente VScroll
OnScrollLineLeft	Manejador de evento "Desplazar línea a la izquierda" (virtual) del control dependiente HScroll
OnScrollLineRight	Manejador de evento "Desplazar línea a la derecha" (virtual) del control dependiente HScroll
Cambiar tamaño	
Rebound	Establece las nuevas coordenadas del control utilizando las coordenadas de la clase CRect

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CWndClient.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ColorBackground

Establece el color de fondo del control.

```
bool  ColorBackground(
   const color  value      // color nuevo
   )
```

Parámetros

value

[in] Nuevo color de fondo del control.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ColorBorder

Establece el color de borde del control.

```
bool  ColorBorder(
   const color  value      // color
   )
```

Parámetros

value

[in] Nuevo color de borde del control.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

BorderType

Establece el tipo de borde del control.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // tipo de borde
   )
```

Parámetros

type

[in] Tipo de borde del control.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

VScrolled (Método Get)

Obtiene la bandera que indica el uso de la barra de desplazamiento vertical.

```
bool  VScrolled()
```

Valor devuelto

true, si la barra de desplazamiento se utiliza, false en caso contrario.

VScrolled (Método Set)

Establece la bandera que indica el uso de la barra de desplazamiento vertical.

```
bool  VScrolled(
   const bool  flag      // bandera
   )
```

Parámetros

flag

[in] Bandera.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

HScrolled (Método Get)

Obtiene la bandera que indica el uso de la barra de desplazamiento horizontal.

```
bool  HScrolled()
```

Valor devuelto

true, si la barra de desplazamiento horizontal se utiliza, false en caso contrario.

HScrolled (Método Set)

Establece la bandera que indica el uso de la barra de desplazamiento horizontal.

```
bool  HScrolled(
   const bool  flag      // bandera
   )
```

Parámetros

flag

[in] Bandera.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateBack

Crea el botón de fondo del control.

```
virtual bool  CreateBack()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateScrollV

Crea la barra de desplazamiento vertical.

```
virtual bool  CreateScrollV()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateScrollH

Crea la barra de desplazamiento horizontal.

```
virtual bool  CreateScrollH()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnResize

El manejador de evento del control "Cambiar tamaño".

```
virtual bool  OnResize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

El manejador de evento del control "VScrollShow" (mostrar barra de desplazamiento).

```
virtual bool  OnVScrollShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

El manejador de evento del control "VScrollHide" (ocultar barra de desplazamiento vertical).

```
virtual bool  OnVScrollHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnHScrollShow

El manejador de evento del control "HScrollShow" (mostrar la barra de desplazamiento horizontal).

```
virtual bool  OnHScrollShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnHScrollHide

El manejador de evento del control "HScrollHide" (esconder la barra de desplazamiento vertical).

```
virtual bool  OnHScrollHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

El manejador de evento del control "Línea de desplazamiento hacia abajo" (vertical).

```
virtual bool  OnScrollLineDown()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

El manejador de evento del control "Línea de desplazamiento hacia arriba" (vertical).

```
virtual bool  OnScrollLineUp()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineLeft

El manejador de evento del control "ScrollLineLeft" (línea de desplazamiento horizontal a la izquierda).

```
virtual bool  OnScrollLineLeft()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineRight

El manejador de evento del control "ScrollLineRight" (línea de desplazamiento horizontal a la derecha).

```
virtual bool  OnScrollLineRight()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El método de la clase base no hace nada y siempre devuelve true.

© 2000-2025, MetaQuotes Ltd.

ReBound

Establece las nuevas coordenadas del control utilizando coordenadas de la clase CRect.

```
void  ReBound(
   const & CRect  rect      // Clase CRect
   )
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Klasse CListView

CListView ist eine Klasse des kombinierten Steuerelements "Liste einblenden".

Beschreibung

Die CListView Klasse ermöglicht Anzeige einer Liste mit Datenelementen.

Deklaration

```
class CListView : public CWndClient
```

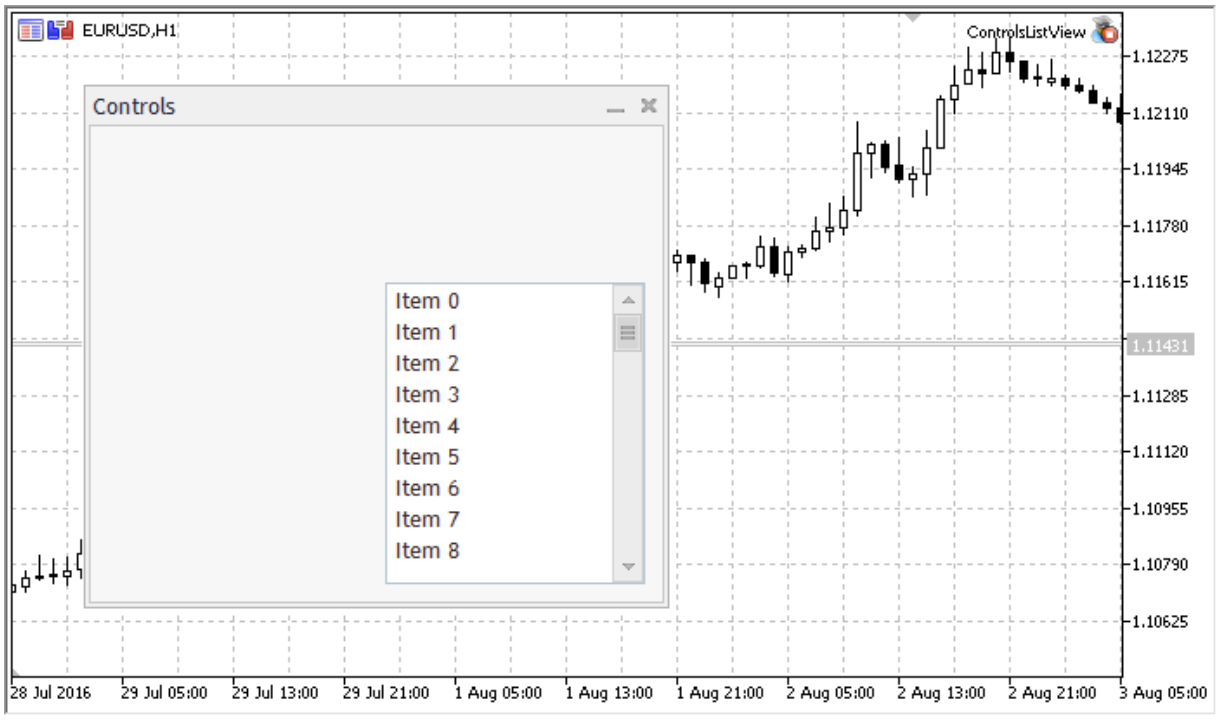
Kopf

```
#include <Controls\ListView.mqh>
```

Jerarquía de herencia

CObject
CWnd
CWndContainer
CWndClient
CListView

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Create	Erstellt ein Steuerelement
Behandlung von Chart-Ereignissen	
OnEvent	Behandelt alle Chart-Ereignisse
Einstellung	
TotalView	Setzt einen Parameter, der die Anzahl der angezeigten Elemente bestimmt
Füllung	
AddItem	Fügt ein Element (Zeile) in die Liste eines Steuerelements hinzu
Daten	
Select	Setzt das aktuelle Listenelement nach dem Index
SelectByText	Setzt das aktuelle Listenelement nach dem Text
SelectByValue	Setzt das aktuelle Listenelement nach dem Wert
Daten (nur lesen)	
Value	Erhält den Wert des aktuellen Listenelements
Unterlegende Steuerelemente	
CreateRow	Erstellt eine "Zeile" des Elements in der Liste
Behandlung von internen Ereignissen	
OnResize	Virtueller Handler des internen Ereignisses "Resize" eines Steuerelements
Behandlung der Ereignisse der unterlegenden Steuerelemente	
OnVScrollShow	Virtueller Handler des internen Ereignisses "Show" des unterlegenden Steuerelements VScroll
OnVScrollHide	Virtueller Handler des internen Ereignisses "Hide" des unterlegenden Steuerelements VScroll
OnScrollLineDown	Virtueller Handler des internen Ereignisses "ScrollLineDown" des unterlegenden Steuerelements VScroll
OnScrollLineUp	Virtueller Handler des internen Ereignisses "ScrollLineUp" des unterlegenden Steuerelements VScroll
OnItemClick	Virtueller Handler des internen Ereignisses "ItemClick" in der angegebenen Zeile eines Steuerelements

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Neuzeichnung	
Redraw	Neuzeichnung eines Steuerelements
RowState	Änderung des Zustands der angegebenen Zeile des Steuerelements
CheckView	Überprüfung der "Sichtbarkeit" der ausgewählten Zeile des Steuerelements

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

Métodos heredados de la clase CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Ein Beispiel für die Erstellung eines Panels mit einer Liste von Werten:

```
//+------------------------------------------------------------------+
//|                                                ControlsListView.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Das Panel der Anzeige und der Dialoge. Demonstration der CListV
#include <Controls\Dialog.mqh>
#include <Controls\ListView.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
```

© 2000-2025, MetaQuotes Ltd.

```
#define CONTROLS_GAP_X                   (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                   (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                     (100)     // size by X coordinate
#define BUTTON_HEIGHT                    (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                      (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                      (150)     // size by X coordinate
#define LIST_HEIGHT                      (179)     // size by Y coordinate
#define RADIO_HEIGHT                     (56)      // size by Y coordinate
#define CHECK_HEIGHT                     (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CListView         m_list_view;                  // CListView Objekt

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateListView(void);
   //--- handlers of the dependent controls events
   void              OnChangeListView(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_list_view,OnChangeListView)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateListView())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "ListView" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateListView(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+GROUP_WIDTH+2*CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+2*CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+LIST_HEIGHT-CONTROLS_GAP_Y;
//--- create
   if(!m_list_view.Create(m_chart_id,m_name+"ListView",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_list_view))
      return(false);
//--- fill out with strings
   for(int i=0;i<16;i++)
      if(!m_list_view.AddItem("Item "+IntegerToString(i)))
         return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeListView(void)
  {
   Comment(__FUNCTION__+" \""+m_list_view.Select()+"\"");
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Kommentare löschen
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CListView.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

TotalView

Establece el número de ítemes mostrados en el control.

```
bool  TotalView(
   const int  value      // ítemes mostrados
   )
```

Parámetros

value

[in] El número de ítemes mostrados en el control.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Nota

El número de ítemes mostrados solo se puede especificar a la vez.

© 2000-2025, MetaQuotes Ltd.

AddItem

Añade un ítem al control.

```
bool  AddItem(
   const string  item,     // texto
   const long    value     // valor
   )
```

Parámetros

item

[in] Texto.

value

[in] Valor de tipo long.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Select

Selecciona el ítem de la lista por medio de su índice.

```
bool  Select(
   const int  index     // índice
   )
```

Parámetros

index

[in] Índice del ítem.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

SelectByText

Selecciona el ítem de la lista por medio de su texto.

```
bool  SelectByText(
   const string  text     // texto
   )
```

Parámetros

text

[in] Texto.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

SelectByValue

Selecciona el ítem de la lista por medio de su valor.

```
bool  SelectByValue(
   const long  value     // valor
   )
```

Parámetros

value

[in] Valor de tipo long.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Value

Obtiene el valor del elemento de la lista.

```
long  Value()
```

Valor devuelto

El valor del elemento de la lista.

© 2000-2025, MetaQuotes Ltd.

CreateRow

Crea una fila del control CListView.

```
bool  CreateRow(
   const int  index     // índice
   )
```

Parámetros

index

[in] Índice del ítem.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnResize

El manejador de evento del control "Cambiar tamaño".

```
virtual bool  OnResize()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

El manejador de evento del control "VScrollShow" (mostrar barra de desplazamiento).

```
virtual bool  OnVScrollShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

El manejador de evento del control "VScrollHide" (ocultar barra de desplazamiento vertical).

```
virtual bool  OnVScrollHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

El manejador de evento del control "Línea de desplazamiento hacia abajo" (vertical).

```
virtual bool  OnScrollLineDown()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

El manejador de evento del control "Línea de desplazamiento hacia arriba" (vertical).

```
virtual bool  OnScrollLineUp()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnItemClick

El manejador de evento del control "Clicar ítem" (clic de ratón sobre una fila).

```
virtual bool  OnItemClick()
   const int    index     // índice de la fila
   )
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Redraw

Redibuja el control.

```
bool  Redraw()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

RowState

Establece el estado de la fila especificada.

```
bool  RowState(
   const int   index      // índice
   const bool  select     // estado
   )
```

Parámetros

index

[in] Índice de la fila.

select

[in] Estado de la fila.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CheckView

Comprueba la "visibilidad" de la fila especificada.

```
bool  CheckView()
```

Valor devuelto

true, si la fila seleccionada es visible, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Klasse CComboBox

CComboBox ist eine Klasse des kombinierten Steuerelements "Feld mit einer Dropdown-Liste".

Beschreibung

Klasse CComboBox ist ein Steuerelement mit einer Dropdown-Liste für die Auswahl eines Werts.

Deklaration

```
class CComboBox : public CWndContainer
```

Kopf

```
#include <Controls\ComboBox.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndContainer
      CComboBox

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

Erstellung	
Create	Erstellt ein Steuerelement

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Behandlung von Chart-Ereignissen	
OnEvent	Behandelt alle Chart-Ereignisse
Füllung	
AddItem	Fügt ein Element (Zeile) in die Liste eines Steuerelements hinzu
Einstellung	
ListViewItems	Setzt die Anzahl der Elemente in der Dropdown-Liste des Steuerelements
Daten	
Select	Setzt das aktuelle Listenelement nach dem Index
SelectByText	Setzt das aktuelle Listenelement nach dem Text
SelectByValue	Setzt das aktuelle Listenelement nach dem Wert
Daten (nur lesen)	
Value	Erhält den Wert des aktuellen Listenelements
Unterlegende Steuerelemente	
CreateEdit	Erstellt ein unterlegendes Steuerelement (Eingabefeld) eines Steuerelements
CreateButton	Erstellt ein unterlegendes Steuerelement (Taste) eines Steuerelements
CreateList	Erstellt ein unterlegendes Steuerelement (Dropdown-Liste) eines Steuerelements
Behandlung der Ereignisse der unterlegenden Steuerelemente	
OnClickEdit	Virtueller Handler des internen Ereignisses "ClickEdit" eines Steuerelements
OnClickButton	Virtueller Handler des internen Ereignisses "ClickButton" eines Steuerelements
OnChangeList	Virtueller Handler des internen Ereignisses "ChangeList" eines Steuerelements
Dropdown-Liste einblenden	
ListShow	Einblendet die Dropdown-Liste eines Steuerelements
ListHide	Ausblendet die Dropdown-Liste eines Steuerelements

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Hide

Ein Beispiel für die Erstellung eines Panel mit dem Steuerelement "Feld mit einer Dropdown-Liste":

```
//+------------------------------------------------------------------+
//|                                              ControlsComboBox.mq5 |
//|                        Copyright 2015, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2015, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CComboBox"
#include <Controls\Dialog.mqh>
#include <Controls\ComboBox.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CComboBox         m_combo_box;;                    // CComboBox object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateComboBox(void);
   //--- handlers of the dependent controls events
   void              OnChangeComboBox(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_combo_box,OnChangeComboBox)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateComboBox())
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "ComboBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateComboBox(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_combo_box.Create(m_chart_id,m_name+"ComboBox",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_combo_box))
      return(false);
//--- fill out with strings
   for(int i=0;i<16;i++)
      if(!m_combo_box.ItemAdd("Item "+IntegerToString(i)))
         return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeComboBox(void)
  {
   Comment(__FUNCTION__+" \""+m_combo_box.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea un nuevo control CComboBox.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

AddItem

Añade un ítem al control.

```
bool  AddItem(
   const string  item,      // texto
   const long    value      // valor
   )
```

Parámetros

item

[in] Texto.

value=0

[in] Valor de tipo long.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ListViewItems

Establece el número de ítemes de la lista del control CComboBox.

```
void  ListViewItems(
   const int    value     // número de ítemes de la lista
   )
```

Parámetros

value

[in] Número de ítemes de la lista.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Select

Selecciona el ítem de la lista por medio de su índice.

```
bool  Select(
   const int  index     // índice
   )
```

Parámetros

index

[in] Índice del ítem.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

SelectByText

Selecciona el ítem de la lista por medio de su texto.

```
bool  SelectByText(
   const string  text     // texto
   )
```

Parámetros

text

[in] Texto del ítem.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

SelectByValue

Selecciona el ítem de la lista por medio de su valor.

```
bool  SelectByValue(
   const long  value     // valor
   )
```

Parámetros

value

[in] Valor de tipo long.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Value

Obtiene el valor del ítem de la lista.

```
long  Value()
```

Valor devuelto

El valor del ítem de la lista.

© 2000-2025, MetaQuotes Ltd.

CreateEdit

Crea el control dependiente (editar).

```
virtual bool  CreateEdit()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateButton

Crea el control dependiente (botón).

```
virtual bool  CreateButton()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateList

Crea el control dependiente (vista de la lista).

```
virtual bool  CreateList()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickEdit

El manejador de evento "Clicar editar" (clic de ratón sobre la edición).

```
virtual bool  OnClickEdit()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickButton

El manejador de evento del control "Clicar botón" (clic de ratón sobre el botón).

```
virtual bool  OnClickButton()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChangeList

El manejador de evento "Cambiar lista" (cambio de la lista).

```
virtual bool  OnChangeList()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ListShow

Muestra la lista de ítemes.

```
virtual bool  ListShow()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ListHide

Esconde la lista de ítemes.

```
virtual bool  ListHide()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Klasse CCheckBox

CCheckBox ist eine Klasse des kombinierten Steuerelements "Schalter mit Fixierung".

Beschreibung

Klasse CCheckBox wird für Erstellung eines Steuerelements, welches Zustand auf den Gegenzustand mit dem Mausklick ändert, verwendet.

Deklaration

```
class CCheckBox : public CWndContainer
```

Kopf

```
#include <Controls\CheckBox.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndContainer
      CCheckBox

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

Erstellung	
Create	Erstellt ein Steuerelement

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Behandlung von Chart-Ereignissen	
OnEvent	Behandelt alle Chart-Ereignisse
Einstellungen	
Text	Erhält/setzt den Erläuterungstext eines Steuerelements
Color	Erhält/setzt die Farbe des Erläuterungstextes eines Steuerelements
Zustand	
Checked	Erhält/setzt den Zustand eines Steuerelements
Daten	
Value	Erhält/setzt den mit dem Steuerelement verbundenen Wert
Unterlegende Steuerelemente	
CreateButton	Erstellt ein unterlegendes Steuerelement (Taste) eines Steuerelements
CreateLabel	Erstellt ein unterlegendes Steuerelement (Erläuterung-Label) eines Steuerelements
Behandlung der Ereignisse der unterlegenden Steuerelemente	
ClickButton	Virtueller Handler des internen Ereignisses "ClickButton" eines Steuerelements
ClickLabel	Virtueller Handler des internen Ereignisses "ClickLabel" eines Steuerelements

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

© 2000-2025, MetaQuotes Ltd.

Ein Beispiel für die Erstellung eines Panels mit dem Steuerelement "Umschalter mit Fixierung":

```
//+------------------------------------------------------------------+
//|                                                ControlsCheckBox.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CCheckBox"
#include <Controls\Dialog.mqh>
#include <Controls\CheckBox.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CCheckBox         m_check_box1;                    // CCheckBox object
   CCheckBox         m_check_box2;                    // CCheckBox object
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
```

© 2000-2025, MetaQuotes Ltd.

```
protected:
   //--- create dependent controls
   bool              CreateCheckBox1(void);
   bool              CreateCheckBox2(void);
   //--- handlers of the dependent controls events
   void              OnChangeCheckBox1(void);
   void              OnChangeCheckBox2(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_box1,OnChangeCheckBox1)
ON_EVENT(ON_CHANGE,m_check_box2,OnChangeCheckBox2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateCheckBox1())
      return(false);
   if(!CreateCheckBox2())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
```

© 2000-2025, MetaQuotes Ltd.

```
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_check_box1.Create(m_chart_id,m_name+"CheckBox1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box1.Text("CheckBox1"))
      return(false);
   if(!m_check_box1.Color(clrBlue))
      return(false);
   if(!Add(m_check_box1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+GROUP_WIDTH+CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_check_box2.Create(m_chart_id,m_name+"CheckBox2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box2.Text("CheckBox2"))
      return(false);
   if(!m_check_box2.Color(clrBlue))
      return(false);
   if(!Add(m_check_box2))
      return(false);
   m_check_box2.Checked(true);
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box2.Checked()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox1(void)
  {
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box1.Checked()));
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox2(void)
  {
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box2.Checked()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea un nuevo control CCheckBox.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Text (Método Get)

Obtiene el texto de la etiqueta asociada al control.

```
string  Text()
```

Valor devuelto

Texto de la etiqueta.

Text (Método Set)

Establece el texto de la etiqueta asociada al control.

```
bool  Text(
   const string  value      // texto
   )
```

Parámetros

value

[in] Texto nuevo de la etiqueta.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Color (Método Get)

Obtiene el color de la etiqueta asociada al control.

```
color  Color()  const
```

Valor devuelto

Color de la etiqueta.

Color (Método Set)

Establece el color de la etiqueta asociada al control.

```
bool  Color(
   const color  value      // color
   )
```

Parámetros

value

[in] Color nuevo de la etiqueta.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Checked (Método Get)

Obtiene el estado del control.

```
bool  Checked()  const
```

Valor devuelto

Estado del control.

Checked (Método Set)

Establece el estado del control.

```
bool  Checked(
   const bool  flag      // estado
   )
```

Parámetros

flag

[in] Estado nuevo.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Value (Método Get)

Obtiene el valor asociado al control.

```
int  Value()  const
```

Valor devuelto

El valor asociado al control.

Value (Método Set)

Establece el valor asociado al control.

```
void  Value(
   const int  value      // valor nuevo
   )
```

Parámetros

value

[in] Valor nuevo.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

CreateButton

Crea el control dependiente (botón).

```
virtual bool  CreateButton()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateLabel

Crea el control dependiente (etiqueta).

```
virtual bool  CreateLabel()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickButton

El manejador de evento del control "Clicar botón" (clic de ratón sobre el botón).

```
virtual bool  OnClickButton()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickLabel

El manejador de evento del control "Clicar etiqueta" (clic de ratón sobre la etiqueta).

```
virtual bool  OnClickLabel()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Klasse CCheckGroup

CCheckGroup ist eine Klasse des kombinierten Steuerelements "Schalter mit einer unabhängigen Fixierung".

Beschreibung

Klasse CCheckGroup wird für Erstellung eines Steuerelements, der einen Satz von Flags anzeigen und bearbeiten erlaubt, verwendet.

Deklaration

```
class CCheckGroup : public CWndClient
```

Kopf

```
#include <Controls\CheckGroup.mqh>
```

Jerarquía de herencia

CObject
CWnd
CWndContainer
CWndClient
CCheckGroup

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Create	Erstellt ein Steuerelement
Behandlung von Chart-Ereignissen	
OnEvent	Behandelt alle Chart-Ereignisse
Füllung	
AddItem	Fügt ein neues Element in der Gruppe hinzu
Daten (nur lesen)	
Value	Erhält den mit dem Steuerelement verbundenen Wert
Unterlegende Steuerelemente	
CreateButton	Erstellt ein neues Element CCheckBox in der Gruppe am angegebenen Index
Behandlung der Ereignisse der unterlegenden Steuerelemente	
OnVScrollShow	Virtueller Handler des internen Ereignisses "Show" des unterlegenden Steuerelements VScroll.
OnVScrollHide	Virtueller Handler des internen Ereignisses "Hide" des unterlegenden Steuerelements VScroll.
OnScrollLineDown	Virtueller Handler des internen Ereignisses "ScrollLineUp" des unterlegenden Steuerelements VScroll.
OnScrollLineUp	Virtueller Handler des internen Ereignisses "ScrollLineDown" des unterlegenden Steuerelements VScroll.
OnChangeItem	Virtueller Handler des internen Ereignisses "ChangeItem" eines Steuerelements
Neuzeichnung	
Redraw	Zeichnet eine Elementgruppe neu
RowState	Ändert den Zustand eines Elements der Gruppe

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Métodos heredados de la clase CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Ein Beispiel für die Erstellung eines Panels mit dem Steuerelement "Umschalter mit einer unabhängigen Fixierung":

```
//+------------------------------------------------------------------+
//|                                            ControlsCheckGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CCheckGroup"
#include <Controls\Dialog.mqh>
#include <Controls\CheckGroup.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
class CControlsDialog : public CAppDialog
  {
private:
   CCheckGroup       m_check_group;                    // CCheckGroup object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateCheckGroup(void);
   //--- handlers of the dependent controls events
   void              OnChangeCheckGroup(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_group,OnChangeCheckGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateCheckGroup())
      return(false);
//--- succeed
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Create the "CheckGroup" element                                  |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckGroup(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+CHECK_HEIGHT;
//--- create
   if(!m_check_group.Create(m_chart_id,m_name+"CheckGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_check_group))
      return(false);
//--- fill out with strings
   for(int i=0;i<5;i++)
      if(!m_check_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_check_group.Check(0,1<<0);
   m_check_group.Check(2,1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(ChartID(),"Controls",0,40,40,380,344))
      return(INIT_FAILED);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea un nuevo control CCheckGroup.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

AddItem

Añade un ítem al control.

```
bool  AddItem(
   const string  item,     // texto
   const long    value     // valor
   )
```

Parámetros

item

[in] Texto.

value=0

[in] Valor de tipo long.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Value

Obtiene el valor asociado al control.

```
long  Value()
```

Valor devuelto

El valor asociado al control.

Nota

El valor depende del estado de todos los ítemes de CCheckGroup.

© 2000-2025, MetaQuotes Ltd.

CreateButton

Crea una nueva instancia de la clase CCheckBox en el índice especificado.

```
bool  CreateButton(
   int  index      // índice
   )
```

Parámetros

index

[in] Índice del nuevo ítem en el CCheckGroup.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

El manejador de evento del control "VScrollShow" (mostrar barra de desplazamiento).

```
virtual bool  OnVScrollShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

El manejador de evento del control "VScrollHide" (ocultar barra de desplazamiento vertical).

```
virtual bool  OnVScrollHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

El manejador de evento del control "Línea de desplazamiento hacia abajo" (vertical).

```
virtual bool  OnScrollLineDown()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

El manejador de evento del control "Línea de desplazamiento hacia arriba" (vertical).

```
virtual bool  OnScrollLineUp()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChangeItem

El manejador de evento "Cambiar ítem".

```
virtual bool  OnChangeItem(
   const int  index      // índice
   )
```

Parámetros

index

[in] Índice del ítem cambiado.

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Redraw

Redibuja el control.

```
bool  Redraw()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

RowState

Establece el estado del ítem especificado.

```
bool  RowState(
   const int   index,      // índice del ítem
   const bool  select      // estado
   )
```

Parámetros

index

[in] Índice del ítem a cambiar.

select

[in] Estado nuevo.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CRadioButton

CRadioButton es una clase del control complejo RadioButton.

Descripción

La clase CRadioButton no se utiliza, se utiliza para la creación de ítemes CRadioGroup.

Declaración

```
class CRadioButton : public CWndContainer
```

Título

```
#include <Controls\RadioButton.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndContainer
      CRadioButton

Métodos de la clase

Creación	
Create	Crea el control
Manejadores de eventos gráficos	
OnEvent	Manejador de todos los eventos gráficos
Propiedades	
Text	Obtiene/Establece la etiqueta de texto asociada al control
Color	Obtiene/Establece el color de la etiqueta de texto asociada al control
Estado	
State	Obtiene/Establece el estado
Controles dependientes	
CreateButton	Crea el botón
CreateLabel	Crea la etiqueta
Manejadores de eventos de controles dependientes	
OnClickButton	Manejador del evento "Clicar botón" (virtual)

© 2000-2025, MetaQuotes Ltd.

Creación	
OnClickLabel	Manejador del evento "Clicar etiqueta" (virtual)

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CRadioButton.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Text (Método Get)

Obtiene el texto de la etiqueta asociada al control.

```
string  Text()
```

Valor devuelto

Texto de la etiqueta.

Text (Método Set)

Establece el texto de la etiqueta asociada al control.

```
bool  Text(
   const string  value      // texto
   )
```

Parámetros

value

[in] Texto nuevo de la etiqueta.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Color (Método Get)

Obtiene el color de la etiqueta asociada al control.

```
color  Color()  const
```

Valor devuelto

Color de la etiqueta.

Color (Método Set)

Establece el color de la etiqueta asociada al control.

```
bool  Color(
   const color  value      // color
   )
```

Parámetros

value

[in] Color nuevo de la etiqueta.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

State (Método Get)

Obtiene el estado del botón.

```
bool  State()  const
```

Valor devuelto

Estado del botón.

State (Método Set)

Establece el estado del botón.

```
bool  State(
   const bool  flag      // bandera
   )
```

Parámetros

flag

[in] Nuevo estado del botón.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateButton

Crea el botón.

```
virtual bool  CreateButton()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateLabel

Crea la etiqueta.

```
virtual bool  CreateLabel()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickButton

El manejador de evento del control "Clicar botón" (clic de ratón).

```
virtual bool  OnClickButton()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickLabel

El manejador de evento del control "Clicar etiqueta" (clic de ratón en la etiqueta).

```
virtual bool  OnClickLabel()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Klasse CRadioGroup

CRadioGroup ist eine Klasse des kombinierten Steuerelements "Schalter mit abhängigen Fixierung".

Beschreibung

Klasse CRadioGroup wird für Erstellung eines Steuerelements, der ein Feld des Enumerationstyps anzeigen und bearbeiten erlaubt.

Deklaration

```
class CRadioGroup : public CWndClient
```

Kopf

```
#include <Controls\RadioGroup.mqh>
```

Jerarquía de herencia

CObject
CWnd
CWndContainer
CWndClient
CRadioGroup

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Create	Erstellt ein Steuerelement
Behandlung von Chart-Ereignissen	
OnEvent	Behandelt alle Chart-Ereignisse
Füllung	
AddItem	Fügt ein neues Gruppenelement hinzu
Daten (nur lesen)	
Value	Erhält den mit dem Zustand eines Steuerelements verbundenen Wert
Unterlegende Steuerelemente	
CreateButton	Erstellt ein neues Element in der Gruppe am angegebenen Index
Behandlung der Ereignisse der unterlegenden Steuerelemente	
OnVScrollShow	Virtueller Handler des internen Ereignisses "Show" des unterlegenden Steuerelements VScroll.
OnVScrollHide	Virtueller Handler des internen Ereignisses "Hide" des unterlegenden Steuerelements VScroll.
OnScrollLineDown	Virtueller Handler des internen Ereignisses "ScrollLineDown" des unterlegenden Steuerelements VScroll
OnScrollLineUp	Virtueller Handler des internen Ereignisses "ScrollLineUp" des unterlegenden Steuerelements VScroll.
OnChangeItem	Virtueller Handler des internen Ereignisses "ChangeItem" eines Steuerelements
Neuzeichnung	
Redraw	Zeichnet eine Elementgruppe neu
RowState	Ändert den Zustand eines Elements der Gruppe
Select	Wählt das aktuelle Element aus

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet,

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Métodos heredados de la clase CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Ein Beispiel für die Erstellung eines Panels mit einer Gruppe von "Radio Buttons":

```
//+------------------------------------------------------------------+
//|                                            ControlsRadioGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Das Panel der Anzeige und der Dialoge. Demonstration der CRadio
#include <Controls\Dialog.mqh>
#include <Controls\RadioGroup.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CRadioGroup       m_radio_group;                    // CRadioGroup Objekt

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateRadioGroup(void);
   //--- handlers of the dependent controls events
   void              OnChangeRadioGroup(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_radio_group,OnChangeRadioGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateRadioGroup())
      return(false);
//--- succeed
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "RadioGroup" element                                  |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateRadioGroup(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+RADIO_HEIGHT;
//--- create
   if(!m_radio_group.Create(m_chart_id,m_name+"RadioGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_radio_group))
      return(false);
//--- fill out with strings
   for(int i=0;i<3;i++)
      if(!m_radio_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_radio_group.Value(1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeRadioGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Kommentare löschen
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CRadioGroup.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

AddItem

Añade un ítem al control.

```
bool  AddItem(
   const string  item,      // texto
   const long    value=0    // valor
   )
```

Parámetros

item

[in] Texto.

value=0

[in] Valor de tipo long.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Value

Obtiene el valor asociado al control.

```
long  Value()
```

Valor devuelto

El valor asociado al control.

Nota

El valor depende del estado de todos los ítemes CRadioButton del control CRadioGroup.

© 2000-2025, MetaQuotes Ltd.

CreateButton

Crea una nueva instancia de la clase CRadioButton en el índice especificado.

```
bool  CreateButton(
   const int  index      // índice
   )
```

Parámetros

index

[in] Índice del nuevo ítem en el CRadioGroup.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

El manejador de evento del control "VScrollShow" (mostrar barra de desplazamiento).

```
virtual bool  OnVScrollShow()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

El manejador de evento del control "VScrollHide" (ocultar barra de desplazamiento vertical).

```
virtual bool  OnVScrollHide()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

El manejador de evento del control "Línea de desplazamiento hacia abajo" (vertical).

```
virtual bool  OnScrollLineDown()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

El manejador de evento del control "Línea de desplazamiento hacia arriba" (vertical).

```
virtual bool  OnScrollLineUp()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChangeItem

El manejador de evento "Cambiar ítem".

```
virtual bool  OnChangeItem(
   const int  index      // índice
   )
```

Parámetros

index

[in] Índice del ítem cambiado.

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Redraw

Redibuja el control.

```
bool  Redraw()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

RowState

Establece el estado del ítem especificado.

```
bool  RowState(
   const int   index,      // índice del ítem
   const bool  select      // estado
   )
```

Parámetros

index

[in] Índice del ítem a cambiar.

select

[in] Estado nuevo.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Select

Selecciona el ítem actual.

```
void  Select(
   const int  index      // índice
   )
```

Parámetros

index

[in] Índice del ítem a seleccionar.

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Klasse CSpinEdit

CSpinEdit ist eine Klasse des kombinierten Steuerelements "Feld von Inkrement-Dekrement".

Beschreibung

Mit der CSpinEdit-Klasse erstellen Sie ein Steuerelement, um den Wert einer Integer-Variable mit den angegebenen Schritten in den festgelegten Grenzen zu bearbeiten.

Deklaration

```
class CSpinEdit : public CWndContainer
```

Kopf

```
#include <Controls\SpinEdit.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndContainer
      CSpinEdit

Das Ergebnis des unten angeführten Codes:



Gruppen der Klassenmethode

Erstellung	
Create	Erstellt ein Steuerelement

© 2000-2025, MetaQuotes Ltd.

Erstellung	
Handler von Chart-Ereignissen	
OnEvent	Behandelt alle Chart-Ereignisse
Einstellung	
MinValue	Erhält/setzt den Minimalwert eines Steuerelements
MaxValue	Erhält/setzt den Maximalwert eines Steuerelements
Zustand	
Value	Erhält/setzt den aktuellen Wert eines Steuerelements
Unterlegende Steuerelemente	
CreateEdit	Erstellt ein unterlegendes Steuerelement CEdit eines Steuerelements
CreateInc	Erstellt eine Taste für Erhöhung des Werts (Inkrement) eines Steuerelements
CreateDec	Erstellt eine Taste für Reduzierung des Werts (Dekrement) eines Steuerelements
Behandlung der Ereignisse der unterlegenden Steuerelemente	
OnClickInc	Virtueller Handler des internen Ereignisses "ClickInc" eines Steuerelements
OnClickDec	Virtueller Handler des internen Ereignisses "ClickDec" eines Steuerelements
Behandlung von internen Ereignissen	
OnChangeValue	Virtueller Handler des internen Ereignisses "ChangeValue" eines Steuerelements

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Ein Beispiel für die Erstellung eines Panels mit dem Scrollen von Werten:

```
//+------------------------------------------------------------------+
//|                                              ControlsSpinEdit.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Das Panel der Anzeige und der Dialoge. Demonstration der CSpinE
#include <Controls\Dialog.mqh>
#include <Controls\SpinEdit.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CSpinEdit         m_spin_edit;                    // CSpinEdit Objekt

public:
```

© 2000-2025, MetaQuotes Ltd.

```
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateSpinEdit(void);
   //--- handlers of the dependent controls events
   void              OnChangeSpinEdit(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
   ON_EVENT(ON_CHANGE,m_spin_edit,OnChangeSpinEdit)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateSpinEdit())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "SpinEdit" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateSpinEdit(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+(BUTTON_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_spin_edit.Create(m_chart_id,m_name+"SpinEdit",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_spin_edit))
      return(false);
   m_spin_edit.MinValue(10);
   m_spin_edit.MaxValue(100);
   m_spin_edit.Value(50);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeSpinEdit(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Kommentare löschen
   Comment("");
//--- destroy dialog
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Crea el nuevo control CSpinEdit.

```
virtual bool  Create(
   const long     chart,      // Identificador gráfico
   const string   name,       // nombre
   const int      subwin,     // subventana gráfica
   const int      x1,         // coordenada x1
   const int      y1,         // coordenada y1
   const int      x2,         // coordenada x2
   const int      y2          // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

MinValue (Método Get)

Obtiene el valor de la propiedad "Valor mínimo" (mínimo valor permitido) del control.

```
int  MinValue()  const
```

Valor devuelto

El valor de la propiedad "Valor mínimo".

MinValue (Método Set)

Establece el valor de la propiedad "Valor mínimo" (mínimo valor permitido) del control.

```
void  MinValue(
   const int  value      // valor nuevo
   )
```

Parámetros

value

[in] Nuevo valor de la propiedad "Valor mínimo".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

MaxValue (Método Get)

Obtiene el valor de la propiedad "Valor máximo" (máximo valor permitido) del control.

```
int  MaxValue()  const
```

Valor devuelto

El valor de la propiedad "Valor máximo".

MaxValue (Método Set)

Establece el valor de la propiedad "Valor máximo" (máximo valor permitido) del control.

```
void  MaxValue(
   const int  value      // valor nuevo
   )
```

Parámetros

value

[in] Nuevo valor de la propiedad "Valor máximo".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Value (Método Get)

Obtiene la propiedad "Valor" (valor actual) del control.

```
int  Value()  const
```

Valor devuelto

La propiedad "Valor".

Value (Método Set)

Establece la propiedad "Valor" (valor actual) del control.

```
void  Value(
   const int  value      // valor
   )
```

Parámetros

value

[in] Nueva propiedad "Valor".

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

CreateEdit

Crea el control dependiente (CEdit).

```
virtual bool  CreateEdit()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateInc

Crea el control dependiente (botón de incremento).

```
virtual bool  CreateInc()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateDec

Crea el botón dependiente (botón de decremento).

```
virtual bool  CreateDec()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickInc

El manejador de evento del control "Clicar incremento" (clic de ratón en el botón de incremento).

```
virtual bool  OnClickInc()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickDec

El manejador de evento del control "Clicar decremento" (clic de ratón en el botón de decremento).

```
virtual bool  OnClickDec()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnChangeValue

El manejador de evento del control "Cambiar valor".

```
virtual bool  OnChangeValue()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CDialog

CDialog es una clase del control complejo Dialog.

Descripción

La clase CDialog combina los controles con varias funciones del grupo.

Declaración

```
class CDialog : public CWndContainer
```

Título

```
#include <Controls\Dialog.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndContainer
      CDialog

Descendientes directos

CAppDialog

Métodos de la clase

Creación	
Create	Crea el control
Manejadores de eventos gráficos	
OnEvent	Manejador de todos los eventos gráficos
Propiedades	
Caption	Obtiene/Establece el valor de la propiedad "Leyenda"
Add	
Add	Añade el control al área cliente
Controles dependientes	
CreateWhiteBorder	Crea el control dependiente (borde blanco)
CreateBackground	Crea el control dependiente (fondo)
CreateCaption	Crea el control dependiente (leyenda)
CreateButtonClose	Crea el control dependiente (cerrar botón)

© 2000-2025, MetaQuotes Ltd.

Creación	
CreateClientArea	Crea el control dependiente (área de cliente)
Manejadores de eventos de controles dependientes	
OnClickCaption	Manejador del evento "Clicar leyenda"
OnClickButtonClose	Manejador del evento "Clicar botón cerrar"
Accede al área de cliente	
ClientAreaVisible	Establece un valor que indica si el área de cliente es visible
ClientAreaLeft	Obtiene la coordenada X de la esquina superior izquierda del área cliente del control
ClientAreaTop	Obtiene la coordenada Y de la esquina superior izquierda del área cliente del control
ClientAreaRight	Obtiene la coordenada X de la esquina inferior derecha del área cliente del control
ClientAreaBottom	Obtiene la coordenada Y de la esquina inferior derecha del área cliente del control
ClientAreaWidth	Obtiene la anchura del área cliente
ClientAreaHeight	Obtiene la altura del área cliente
Manejadores de eventos de arrastre	
OnDialogDragStart	Manejador de evento "Iniciar arrastre de diálogo" (virtual)
OnDialogDragProcess	Manejador de evento "Procesar arrastre de diálogo" (virtual)
OnDialogDragEnd	Manejador de evento "Finalizar arrastre de diálogo" (virtual)

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos heredados de la clase CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

© 2000-2025, MetaQuotes Ltd.

Create

Crea un nuevo control CDialog.

```
virtual bool  Create(
   const long    chart,      // Identificador gráfico
   const string  name,       // nombre
   const int     subwin,     // subventana gráfica
   const int     x1,         // coordenada x1
   const int     y1,         // coordenada y1
   const int     x2,         // coordenada x2
   const int     y2          // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Caption (Método Get)

Obtiene la propiedad "Leyenda" del control CDialog.

```
string  MinValue()  const
```

Valor devuelto

La propiedad "Leyenda".

Caption (Método Set)

Establece la propiedad "Leyenda" del control CDialog.

```
bool  Caption(
   const string  text      // texto
   )
```

Parámetros

text

[in] Nuevo valor de la propiedad "Leyenda".

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Add

Añade el control al área cliente por medio de un puntero.

```
bool  Add(
   CWnd   *control,         // puntero
   )
```

Parámetros

control

[in] Puntero al control.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

Add

Añade el control al área cliente por referencia.

```
bool  Add(
   CWnd   &control,         // referencia
   )
```

Parámetros

control

[in] Referencia al control.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateWhiteBorder

Crea el control dependiente (borde blanco).

```
virtual bool  CreateWhiteBorder()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateBackground

Crea el control dependiente (fondo).

```
virtual bool  CreateBackground()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateCaption

Crea el control dependiente (leyenda).

```
virtual bool  CreateCaption()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateButtonClose

Crea el control dependiente (cerrar botón)

```
virtual bool  CreateButtonClose()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateClientArea

Crea el control dependiente (área cliente).

```
virtual bool  CreateClientArea()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickCaption

El manejador de evento "Clicar leyenda".

```
virtual bool  OnClickCaption()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

OnClickButtonClose

El manejador de evento "Clicar botón cerrar".

```
virtual bool  OnClickButtonClose()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ClientAreaVisible

Establece una bandera que indica si el área cliente es visible.

```
bool  ClientAreaVisible(
   const bool  visible      // bandera de visibilidad
   )
```

Parámetros

visible

[in] Bandera de visibilidad.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ClientAreaLeft

Obtiene la coordenada X de la esquina superior izquierda del área cliente del control.

```
int  ClientAreaLeft()
```

Valor devuelto

La coordenada X de la esquina superior izquierda del área cliente del control..

© 2000-2025, MetaQuotes Ltd.

ClientAreaTop

Obtiene la coordenada Y de la esquina superior izquierda del área cliente del control.

```
int  ClientAreaTop()
```

Valor devuelto

La coordenada Y de la esquina superior izquierda del área cliente del control.

© 2000-2025, MetaQuotes Ltd.

ClientAreaRight

Obtiene la coordenada X de la esquina inferior derecha del área cliente del control.

```
int  ClientAreaTop()
```

Valor devuelto

La coordenada X de la esquina inferior derecha del área cliente del control.

© 2000-2025, MetaQuotes Ltd.

ClientAreaBottom

Obtiene la coordenada Y de la esquina inferior derecha del área cliente del control.

```
int  ClientAreaBottom()
```

Valor devuelto

La coordenada Y de la esquina inferior derecha del área cliente del control.

© 2000-2025, MetaQuotes Ltd.

ClientAreaWidth

Obtiene la anchura del área cliente del control.

```
int  ClientAreaWidth()
```

Valor devuelto

La anchura del área cliente.

© 2000-2025, MetaQuotes Ltd.

ClientAreaHeight

Obtiene la altura del área cliente del control.

```
int  ClientAreaHeight()
```

Valor devuelto

La altura del área cliente del control.

© 2000-2025, MetaQuotes Ltd.

OnDialogDragStart

El manejador de evento del control "Inicio de arrastre de diálogo".

```
virtual bool  OnDialogDragStart()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Inicio de arrastre de diálogo" ocurre cuando se empieza a arrastrar el control.

© 2000-2025, MetaQuotes Ltd.

OnDialogDragProcess

El manejador de evento del control "Proceso de arrastre de diálogo".

```
virtual bool  OnDialogDragProcess()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Proceso de arrastre de diálogo" ocurre cuando se arrastra el control.

© 2000-2025, MetaQuotes Ltd.

OnDialogDragEnd

El manejador de evento del control "Finalizar el arrastre del diálogo".

```
virtual bool  OnDialogDragEnd()
```

Valor devuelto

true si el evento se ha procesado, false en caso contrario.

Nota

El evento "Finalizar el arrastre del diálogo" ocurre cuando se termina de arrastrar el control.

© 2000-2025, MetaQuotes Ltd.

CAppDialog

CAppDialog es una clase del control complejo Application Dialog (con controles dependientes).

Descripción

La clase CAppDialog combina los controles con varias funciones dentro de un programa MQL5.

Declaración

```
class CAppDialog : public CDialog
```

Título

```
#include <Controls\Dialog.mqh>
```

Jerarquía de herencia

CObject
  CWnd
    CWndContainer
      CDialog
        CAppDialog

Métodos de la clase

Crear y destruir	
Create	Crea el control
Destroy	Destruye el control
Procesamiento de eventos	
OnEvent	Manejador de todos los eventos gráficos
Ejecución	
Run	Ejecuta el control
Procesamiento de eventos gráficos	
ChartEvent	Manejador de todos los eventos gráficos
Configuraciones	
Minimized	Establece un valor indicando si el control está minimizado
Guardar/Cargar	
IniFileSave	Guarda el estado de control en el archivo
IniFileLoad	Carga el estado de control a partir del archivo

© 2000-2025, MetaQuotes Ltd.

Crear y destruir	
IniFileName	Establece el nombre del archivo para cargar/guardar el estado de control
IniFileExt	Establece la extensión del archivo para cargar/guardar el estado de control
Inicialización	
CreateCommon	Método común de inicialización
CreateExpert	Método de inicialización para trabajar con Asesores Expertos
CreateIndicator	Método de inicialización para trabajar con indicadores
Controles dependientes	
CreateButtonMinMax	Crea controles dependientes (botones de minimizar/maximizar)
Manejadores de eventos de controles dependientes	
OnClickButtonClose	Manejador del evento (virtual) "Clic de botón cerrar"
OnClickButtonMinMax	Manejador del evento (virtual) "Clic de botón MinMax"
Eventos externos	
OnAnotherApplicationClose	Manejador de eventos externos (virtual)
Métodos	
Rebound	Establece las nuevas coordenadas del control utilizando las coordenadas de la clase CRect
Minimize	Muestra el control en estado minimizado
Maximize	Muestra el control en estado (restaurado) maximizado
CreateInstanceId	Crea un identificador único para los nombres de los objetos de control
ProgramName	Obtiene el nombre del programa MQL5
SubwinOff	Obtiene el desplazamiento Y de la subventana de control

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2025, MetaQuotes Ltd.

Métodos heredados de la clase CObject

Prev, Prev, Next, Next, Type, Compare

Métodos heredados de la clase CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Métodos heredados de la clase CDialog

Caption, Caption, Add, Add

© 2000-2025, MetaQuotes Ltd.

Create

Crea un nuevo control CAppDialog.

```
virtual bool  Create(
   const long     chart,       // Identificador gráfico
   const string   name,        // nombre
   const int      subwin,      // subventana gráfica
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Destroy

Método de desinicialización del elemento de control CAppDialog.

```
virtual void  Destroy(
   const int  reason=REASON_PROGRAM    // código del motivo
)
```

Parámetros

reason

[in] Código del motivo de la desinicialización. Por defecto, se indica REASON_PROGRAM.

Valor retornado

No.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Manejador de evento gráfico.

```
virtual bool  OnEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Run

Ejecuta el control.

```
bool  Run()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

ChartEvent

Manejador de evento gráfico.

```
virtual bool  ChartEvent(
   const int      id,          // Identificador
   const long&    lparam,      // parámetro evento de tipo long
   const double&  dparam,      // parámetro evento de tipo double
   const string&  sparam       // parámetro evento de tipo string
   )
```

Parámetros

id

[in] Identificador del evento.

lparam

[in] Parámetro evento de tipo long, pasado por referencia.

dparam

[in] Parámetro evento de tipo double, pasado por referencia.

sparam

[in] Parámetro evento de tipo string, pasado por referencia.

Valor devuelto

true - si el evento se ha procesado, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Minimized

Establece el valor de la propiedad "Minimizado" del control.

```
bool  Minimized(
   const bool  flag      // estado
   )
```

Parámetros

flag

[in] Estado nuevo.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

IniFileSave

Guarda el estado del control en el archivo.

```
void  IniFileSave()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

IniFileLoad

Carga el estado del control a partir del archivo.

```
void  IniFileLoad()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

IniFileName

Establece el nombre del archivo para cargar/guardar el estado del control.

```
virtual string  IniFileName()  const
```

Valor devuelto

Nombre del archivo para cargar/guardar el estado del control.

Nota

El nombre del archivo incluye el nombre del Asesor Experto/indicador y el símbolo de trabajo, donde se carga el programa MQL5.

© 2000-2025, MetaQuotes Ltd.

IniFileExt

Establece la extensión del archivo para cargar/guardar el estado del control.

```
virtual string  IniFileExt()  const
```

Valor devuelto

Extensión del archivo, utilizada para cargar/guardar el estado del control.

© 2000-2025, MetaQuotes Ltd.

CreateCommon

Método de inicialización común.

```
bool  CreateCommon(
   const long    chart,       // Identificador gráfico
   const string  name,        // nombre
   const int     subwin,      // subventana gráfica
   )
```

Parámetros

chart

[in] Identificador gráfico.

name

[in] Nombre único del control.

subwin

[in] Subventana gráfica.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateExpert

Método de inicialización para trabajar con Asesores Expertos.

```
bool  CreateExpert(
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateIndicator

Método de inicialización para trabajar con indicadores.

```
bool  CreateIndicator(
   const int      x1,          // coordenada x1
   const int      y1,          // coordenada y1
   const int      x2,          // coordenada x2
   const int      y2           // coordenada y2
   )
```

Parámetros

x1

[in] Coordenada X de la esquina superior izquierda.

y1

[in] Coordenada Y de la esquina superior izquierda.

x2

[in] Coordenada X de la esquina inferior derecha.

y2

[in] Coordenada Y de la esquina inferior derecha.

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateButtonMinMax

Crea controles dependientes (botones minimizar/maximizar).

```
virtual void  CreateButtonMinMax()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

OnClickButtonClose

Manejador de evento del control "Clic de botón cerrar" (clic de ratón en el botón cerrar).

```
virtual void  OnClickButtonClose()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

OnClickButtonMinMax

Manejador de evento del control "Clic de botón MinMax" (clic de ratón en el botón minimizar/maximizar).

```
virtual void  OnClickButtonClose()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

OnAnotherApplicationClose

Manejador de eventos externos.

```
virtual void  OnAnotherApplicationClose()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Rebound

Establece las nuevas coordenadas del control utilizando coordenadas de la clase CRect.

```
bool  Rebound(
   const & CRect  rect      // Clase CRect
   )
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Minimize

Muestra el control en el estado minimizado.

```
virtual void  Minimize()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

Maximize

Muestra el control en el estado maximizado (restaurado).

```
virtual void  Maximize()
```

Valor devuelto

true si se ejecuta correctamente, false en caso contrario.

© 2000-2025, MetaQuotes Ltd.

CreateInstanceId

Crea un identificador único para los nombres de los objetos de control.

```
string  CreateInstanceId()
```

Valor devuelto

Prefijo de los nombres de los objetos.

© 2000-2025, MetaQuotes Ltd.

ProgramName

Obtiene el nombre del programa MQL5.

```
string  ProgramName()
```

Valor devuelto

Nombre del programa MQL5.

© 2000-2025, MetaQuotes Ltd.

SubwinOff

Obtiene el desplazamiento Y de la subventana de control.

```
void  SubwinOff()
```

Valor devuelto

Ninguno.

© 2000-2025, MetaQuotes Ltd.

Paso de la versión MQL4

El lenguaje MQL5 es un lenguaje desarrollado de su antecesor MQL4, en el cual está escrita una gran cantidad de indicadores, scripts y Asesores Expertos. A pesar de que el lenguaje nuevo es compatible al máximo con la versión anterior, existen unas diferencias entre ellos. Y al pasar los programas hay que saber estas diferencias.

En este apartado está reunida la información cuyo objetivo es facilitar a los programadores que conocen bien Mql4 la adaptación de sus códigos al nuevo lenguaje MQL5.

Ante todo cabe señalar que:

- No hay funciones start(), init() y deinit();
- Cantidad de buffers de indicador no está limitada;
- dll se carga una vez cargado un Asesor Experto (o cualquier otro programa mql5);
- Comprobación reducida de condiciones lógicas;
- Al salir fuera de los límites de un array la ejecución en curso se detiene (de una forma crítica - con mensaje del error);
- Prioridad de operaciones igual que en C++;
- Funciona la conversión implícita de tipos (incluso de cadena a un número);
- Variables locales no se inicializan automáticamente (salvo las cadenas);
- Arrays locales comunes se borran automáticamente.

Funciones especiales init, start y deinit

En el lenguaje MQL4 había sólo tres funciones predefinidas que podían estar presentes en el código de un indicador, Asesor Experto o script (sin tener en cuenta los archivos incluidos *.mqh y archivos de bibliotecas). En MQL5 no hay estas funciones, pero hay sus análogos. En la tabla de abajo se muestra una correspondencia aproximada de estas funciones.

MQL4	MQL5
init	OnInit
start	OnStart
deinit	OnDeinit

Las funciones OnInit y OnDeinit desempeñan el mismo papel que las funciones init y deinit en MQL4, es decir, están predestinadas para colocar el código que debe ejecutarse con la inicialización y deinicialización de los programas mql5. Se puede simplemente renombrar estas funciones de una manera correspondiente, o bien dejarlas tal como están pero hay que añadir la llamada a éstas en los sitios correspondientes.

Ejemplo:

```
void OnInit()
  {
//--- llamamos a la función al inicializar
   init();
```

© 2000-2025, MetaQuotes Ltd.

```
  }
void OnDeinit(const int reason)
  {
//--- llamamos a la función al deinicializar
   deinit();
//---
  }
```

La función start se cambia por OnStart sólo en los scripts, para un Asesor Experto o un indicador hace falta renombrarla con OnTick y OnCalculate respectivamente. El código que tiene que ser ejecutado durante el funcionamiento de un programa mql5, hay que colocarlo precisamente en estas tres funciones:

programa mql5	función principal
script	OnStart
indicador	OnCalculate
Asesor Experto	OnTick

Si el código del indicador o script no contiene la función principal o el nombre de ésta se diferencia del requerido, la llamada a esta función no se realiza. Es decir, si en el código principal de un script no hay función OnStart, este código va a ser compilado como un Asesor Experto.

Si en el código de un indicador falta la función OnCalculate, la compilación de este indicador no es posible.

Variables predefinidas

En MQL5 no hay variables predefinidas como Ask, Bid, Bars. Las variables Digits y Point se han cambiado un poco tal como se muestra en la tabla.

MQL4	MQL5
Digits	_Digits
Point	_Point
	_LastError
	_Period
	_Symbol
	_StopFlag
	_UninitReason

Acceso a series temporales

© 2000-2025, MetaQuotes Ltd.

En MQL5 no hay series temporales predefinidas Open[], High[], Low[], Close[], Volume[] y Time[]. Ahora se puede establecer independientemente la profundidad necesaria de series temporales a través de las correspondientes funciones para acceder a las series temporales.

Asesores Expertos

En MQL5 los Asesores Expertos no requieren la presencia obligatoria de la función-manejador del evento de llegada de un nuevo tick OnTick, como ha sido en MQL4 (en MQL4 la función start se invoca para ser ejecutada cuando llega un nuevo tick), porque ahora en MQL5 los Asesores Expertos pueden contener las funciones-manejadores predefinidas de varios tipos de eventos:

- OnTick – recepción de un nuevo tick;
- OnTimer – evento de temporizador;
- OnTrade - evento comercial;
- OnChartEvent – evento de introducción por el teclado o ratón, evento de movimiento de un objeto gráfico, evento del fin de edición del texto en la casilla de introducción del objeto LabelEdit;
- OnBookEvent – evento del cambio del estado de profundidad de mercado (Depth of Market).

Indicadores personalizados

En MQL4 la cantidad de los buffers de indicador está limitada y no puede superar 8. En MQL5 tal limitación no existe pero hay que recordar que cada búfer de indicadores requiere un cierto volumen de memoria operativa para su ubicación en el terminal, por eso a pesar de todo no hace falta abusar de esta posibilidad.

Además, en MQL4 había sólo 6 tipos de representar gráficamente un indicador personalizado, en MQL5 ahora hay 18 estilos de dibujo. Y aunque los nombres de los tipos de dibujar no se hayan cambiado, la ideología de representación gráfica de los indicadores se ha cambiado considerablemente.

La dirección de la indexación en los buffers de indicadores también se diferencia. Por defecto, en MQL5 todos los búfers se comportan como los arrays comunes, es decir, un elemento con el índice 0 es el más antiguo de la historia, con el aumento del índice nos movemos de los datos más antiguos a los recientes.

La única función para trabajar con indicadores personalizados que se ha conservado de MQL4 es la SetIndexBuffer. Pero su llamada también se ha cambiado, ahora tenemos que especificar tipo de datos que van a guardarse en el array vinculado con el búfer de indicadores.

A parte de eso, las propiedades de los indicadores personalizados se han cambiado y se han extendido. Han sido agregadas nuevas funciones de acceso a series temporales, por eso hay que reconsiderar completamente todo el algoritmo de cálculo.

Objetos gráficos

El número de objetos gráficos en MQL5 se ha incrementado de una manera significativa, sin embargo, ha aparecido una restricción, es que no se puede usar las funciones de trabajo con objetos gráficos en los indicadores personalizados.

Además, el posicionamiento de objetos gráficos en el tiempo ahora es posible con una precisión de hasta un segundo en un gráfico de cualquier período, ahora no se realiza el redondeo de puntos de

© 2000-2025, MetaQuotes Ltd.

anclaje de objetos gráficos con precisión de tiempo de apertura del bar en el actual gráfico de precios.

Para los objetos Arrow, Text y Label ha aparecido la posibilidad de indicar el modo de anclaje, y para los objetos Label, Button, Chart, Bitmap Label y Edit se puede determinar la esquina del gráfico a la que está anclado el objeto.

© 2000-2025, MetaQuotes Ltd.

Lista de Funciones MQL5

Todas las funciones MQL5 en orden alfabético.

Función	Acción	Sección
AccountInfoDouble	Devuelve el valor del tipo double de la propiedad correspondiente de la cuenta	Información de cuenta
AccountInfoInteger	Devuelve el valor del tipo de números enteros (bool,int o long) de la propiedad correspondiente de la cuenta	Información de cuenta
AccountInfoString	Devuelve el valor del tipo string de la propiedad correspondiente de la cuenta	Información de cuenta
acos	Devuelve el valor de arcocoseno de (x) en radianes	Funciones matemáticas
Alert	Muestra el mensaje en una ventana separada	Funciones comunes
ArrayBsearch	Busca el valor especificado en el array numérico multidimensional ordenado en orden ascendente	Operaciones con arrays
ArrayCompare	Devuelve el resultado de comparación de dos arrays de tipos simples o estructuras personalizadas que no tienen objetos complejos	Operaciones con arrays
ArrayCopy	Copia un array al otro	Operaciones con arrays
ArrayFill	Llena un array numérico con valor especificado	Operaciones con arrays
ArrayFree	Deja libre el búfer de cualquier array dinámico y pone el tamaño de la dimensión cero a 0.	Operaciones con arrays
ArrayGetAsSeries	Comprueba la dirección de indexación de un array	Operaciones con arrays
ArrayInitialize	Pone todos los elementos de un array numérico en el mismo valor	Operaciones con arrays
ArrayIsDynamic	Comprueba si un array es dinámico	Operaciones con arrays
ArrayIsSeries	Comprueba si un array es una serie temporal	Operaciones con arrays

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
ArrayMaximum	Busca el elemento máximo en la primera dimensión del array numérico multidimensional	Operaciones con arrays
ArrayMinimum	Busca el elemento mínimo en la primera dimensión del array numérico multidimensional	Operaciones con arrays
ArrayRange	Devuelve el número de elementos en la dimensión especificada del array	Operaciones con arrays
ArrayResize	Fija nuevo tamaño en la primera dimensión del array	Operaciones con arrays
ArraySetAsSeries	Establece la dirección de indexación en el array	Operaciones con arrays
ArraySize	Devuelve el número de elementos en el array	Operaciones con arrays
ArraySort	Ordena el array numérico multidimensional por orden crecientes de los valores en la primera dimensión	Operaciones con arrays
ArrayPrint	Muestra, en el registro, una matriz de tipo o estructura simple	Operaciones con arrays
ArrayInsert	Inserta en la matriz-receptor el número indicado de elementos, comenzando por el índice establecido	Operaciones con arrays
ArrayRemove	Elimina de la matriz el número indicado de elementos, comenzando por el índice indicado	Operaciones con arrays
ArrayReverse	Invierte en la matriz el número indicado de elementos, comenzando por el índice indicado	Operaciones con arrays
ArraySwap	Intercambia el contenido de dos matrices dinámicas del mismo tipo	Operaciones con arrays
Bars	Devuelve la cantidad de barras en el historial por símbolo y período correspondientes	Acceso a las series temporales y a los datos de indicadores
BarsCalculated	Devuelve la cantidad de datos calculados en el búfer de indicadores o -1 en caso del error	Acceso a las series temporales y a los datos de indicadores

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
	(los datos aún no están calculados)	
CalendarCountryById	Obtiene la descripción del país según su identificador	Calendario económico
CalendarEventById	Obtiene la descripción del evento según su identificador	Calendario económico
CalendarValueById	Obtiene la descripción del valor del evento según su identificador	Calendario económico
CalendarCountries	Obtiene la matriz de las descripciones de los países disponibles en el Calendario	Calendario económico
CalendarEventByCountry	Obtiene la matriz de las descripciones de todos los eventos disponibles en el Calendario, según el código del país indicado	Calendario económico
CalendarEventByCurrency	Obtiene la matriz de las descripciones de todos los eventos disponibles en el Calendario, según la divisa indicada	Calendario económico
CalendarValueHistoryByEvent	Obtiene la matriz de valores de todos los eventos en el intervalo temporal indicado según el identificador del evento	Calendario económico
CalendarValueHistory	Obtiene la matriz de valores de todos los eventos en el intervalo temporal indicado con filtro de país y/o divisa	Calendario económico
CalendarValueLastByEvent	Obtiene la matriz de valores de un evento según su ID desde el momento en que se creó la base de datos del Calendario con el change_id indicado	Calendario económico
CalendarValueLast	Obtiene la matriz de valores de todos los eventos con filtrado por país y/o divisa desde el momento en que se creó la base de datos del Calendario con el change_id indicado	Calendario económico
ceil	Devuelve el valor numérico entero más cercano desde arriba	Funciones matemáticas

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
CharArrayToString	Conversión del código del símbolo (ansi) a una cadena de un carácter	Conversión de datos
ChartApplyTemplate	Aplica al gráfico especificado una plantilla desde un archivo especificado	Operaciones con gráficos
ChartClose	Cierra un gráfico especificado	Operaciones con gráficos
ChartFirst	Devuelve el identificador del primer gráfico del terminal de cliente	Operaciones con gráficos
ChartGetDouble	Devuelve el valor de una propiedad correspondiente del gráfico especificado	Operaciones con gráficos
ChartGetInteger	Devuelve el valor del tipo entero de una propiedad correspondiente del gráfico especificado	Operaciones con gráficos
ChartGetString	Devuelve el valor literal de una propiedad correspondiente del gráfico especificado	Operaciones con gráficos
ChartID	Devuelve el identificador del gráfico corriente	Operaciones con gráficos
ChartIndicatorAdd	Añade un indicador con el manejador (handle) especificado a la ventana del gráfico especificado	Operaciones con gráficos
ChartIndicatorDelete	Quita el indicador con el nombre especificado de la ventana del gráfico especificada	Operaciones con gráficos
ChartIndicatorGet	Devuelve el manejador del indicador con el nombre corto especificado en la ventana del gráfico especificada	Operaciones con gráficos
ChartIndicatorName	Devuelve el nombre breve del indicador según su número en la lista de indicadores en la determinada ventana del gráfico	Operaciones con gráficos
ChartIndicatorsTotal	Devuelve el número de todos los indicadores vinculados con la determinada ventana del gráfico	Operaciones con gráficos
ChartNavigate	Desplaza el gráfico especificado a una cantidad de barras	Operaciones con gráficos

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
	especificada respecto a la posición del gráfico indicada	
ChartNext	Devuelve el identificador del gráfico que sigue después del especificado	Operaciones con gráficos
ChartOpen	Abre un gráfico nuevo con un símbolo y período especificados	Operaciones con gráficos
CharToString	Conversión del código del símbolo a una cadena de un carácter	Conversión de datos
ChartPeriod	Devuelve el valor del período del gráfico especificado	Operaciones con gráficos
ChartPriceOnDropped	Devuelve la coordinada de precios que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón	Operaciones con gráficos
ChartRedraw	Activa el redibujo forzado de un gráfico especificado	Operaciones con gráficos
ChartSaveTemplate	Guarda los ajustes actuales del gráfico en una plantilla con el nombre especificado	Operaciones con gráficos
ChartScreenShot	Hace un screenshot del gráfico especificado en formato GIF, PNG o BMP, dependiendo de la extensión especificada	Operaciones con gráficos
ChartSetDouble	Establece el valor del tipo double de una propiedad correspondiente del gráfico especificado	Operaciones con gráficos
ChartSetInteger	Establece el valor del tipo entero (datetime, int, color, bool o char) de una propiedad correspondiente del gráfico especificado	Operaciones con gráficos
ChartSetString	Establece el valor del tipo string de una propiedad correspondiente del gráfico especificado	Operaciones con gráficos
ChartSetSymbolPeriod	Cambia el valor del símbolo y período del gráfico especificado	Operaciones con gráficos
ChartSymbol	Devuelve el nombre del símbolo del gráfico especificado	Operaciones con gráficos
ChartTimeOnDropped	Devuelve la coordinada de tiempo que corresponde al punto al que el	Operaciones con gráficos

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
	Asesor Experto o script ha sido arrastrado con el ratón	
ChartTimePriceToXY	Convierte las coordenadas del gráfico desde la representación hora/precio a las coordinadas en el eje X y Y	Operaciones con gráficos
ChartWindowFind	Devuelve el número de una subventana en la que se encuentra el indicador	Operaciones con gráficos
ChartWindowOnDropped	Devuelve el número de la subventana del gráfico a la que el Asesor Experto, script, objeto o indicador ha sido arrastrado con el ratón	Operaciones con gráficos
ChartXOnDropped	Devuelve la coordinada del eje de X que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón	Operaciones con gráficos
ChartXYToTimePrice	Convierte las coordenadas X y Y del gráfico a los valores hora y precio	Operaciones con gráficos
ChartYOnDropped	Devuelve la coordinada del eje de Y que corresponde al punto al que el Asesor Experto o script ha sido arrastrado con el ratón	Operaciones con gráficos
CheckPointer	Devuelve el tipo del puntero a objeto	Funciones comunes
CLBufferCreate	Crea el búfer OpenCL	Trabajo con OpenCL
CLBufferFree	Elimina el búfer OpenCL	Trabajo con OpenCL
CLBufferRead	Lee el búfer OpenCL en un array	Trabajo con OpenCL
CLBufferWrite	Escribe un array en el búfer OpenCL	Trabajo con OpenCL
CLContextCreate	Crea el contexto OpenCL	Trabajo con OpenCL
CLContextFree	Elimina el contexto OpenCL	Trabajo con OpenCL
CLExecute	Ejecuta un programa OpenCL	Trabajo con OpenCL
CLGetDeviceInfo	Recibe la propiedad del dispositivo desde el controlador de dispositivo OpenCL	Trabajo con OpenCL

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
CLGetInfoInteger	Devuelve el valor de una propiedad de números enteros para el objeto o dispositivo OpenCL	Trabajo con OpenCL
CLHandleType	Devuelve el tipo de manejador OpenCL como un valor de la enumeración ENUM_OPENCL_HANDLE_TYPE	Trabajo con OpenCL
CLKernelCreate	Crea la función del arranque OpenCL	Trabajo con OpenCL
CLKernelFree	Elimina la función del arranque OpenCL	Trabajo con OpenCL
CLProgramCreate	Crea un programa OpenCL desde el código fuente	Trabajo con OpenCL
CLProgramFree	Elimina un programa OpenCL	Trabajo con OpenCL
CLSetKernelArg	Establece un parámetro para la función OpenCL	Trabajo con OpenCL
CLSetKernelArgMem	Establece el búfer OpenCL como el parámetro de la función OpenCL	Trabajo con OpenCL
ColorToARGB	Conversión del tipo color al tipo uint para conseguir la representación del color ARGB.	Conversión de datos
ColorToString	Conversión de valores de colores a una cadena del tipo "R,G,B"	Conversión de datos
Comment	Muestra el mensaje en la esquina superior izquierda del gráfico de precios	Funciones comunes
CopyBuffer	Recibe en el array los datos de un búfer especificado desde un indicador especificado	Acceso a las series temporales y a los datos de indicadores
CopyClose	Recibe en un array los datos históricos sobre el precio de cierre de barras para un símbolo y período especificados	Acceso a las series temporales y a los datos de indicadores
CopyHigh	Recibe en un array los datos históricos sobre el precio máximo de barras para un símbolo y período especificados	Acceso a las series temporales y a los datos de indicadores
CopyLow	Recibe en un array los datos históricos sobre el precio mínimo	Acceso a las series temporales y a los datos de

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
	de barras para un símbolo y período especificados	indicadores
CopyOpen	Recibe en un array los datos históricos sobre el precio de apertura de barras para un símbolo y período especificados	Acceso a las series temporales y a los datos de indicadores
CopyRates	Recibe en un array los datos históricos de la estructura Rates para un símbolo y período especificados	Acceso a las series temporales y a los datos de indicadores
CopyRealVolume	Recibe en un array los datos históricos sobre volúmenes comerciales para un símbolo y período especificados	Acceso a las series temporales y a los datos de indicadores
CopySpread	Recibe en un array los datos históricos sobre los spreads para un símbolo y período especificados	Acceso a las series temporales y a los datos de indicadores
CopyTicks	Recibe en un array los ticks acumulados por el terminal durante la sesión actual	Acceso a las series temporales y a los datos de indicadores
CopyTickVolume	Recibe en un array los datos históricos sobre volúmenes de tick para un símbolo y período especificados	Acceso a las series temporales y a los datos de indicadores
CopyTime	Recibe en un array los datos históricos sobre el tiempo de apertura de barras para un símbolo y período especificados	Acceso a las series temporales y a los datos de indicadores
cos	Devuelve el coseno de un número	Funciones matemáticas
CryptDecode	Conversión inversa de los datos del array	Funciones comunes
CryptEncode	Convierte los datos del array fuente en el array receptor según el método especificado	Funciones comunes
CustomSymbolCreate	Crea un símbolo personalizado con el nombre indicado en el grupo indicado	Símbolos personalizados
CustomSymbolDelete	Elimina el símbolo personalizado con el nombre indicado	Símbolos personalizados

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
CustomSymbolSetInteger	Establece para el símbolo personalizado el valor de propiedad de tipo de número entero	Símbolos personalizados
CustomSymbolSetDouble	Establece para el símbolo personalizado el valor de propiedad de tipo real	Símbolos personalizados
CustomSymbolSetString	Establece para el símbolo personalizado el valor de propiedad de tipo string	Símbolos personalizados
CustomSymbolSetMarginRate	Establece para el símbolo personalizado los coeficientes del margen de carga dependiendo del tipo y la dirección de la orden	Símbolos personalizados
CustomSymbolSetSessionQuote	Establece la hora de comienzo y finalización de la sesión de cotizaciones para los símbolos y el día de la semana indicados	Símbolos personalizados
CustomSymbolSetSessionTrade	Establece la hora de comienzo y finalización de la sesión de comercial para los símbolos y el día de la semana indicados	Símbolos personalizados
CustomRatesDelete	Elimina todas las barras en el intervalo temporal indicado de la historia de precios del instrumento personalizado	Símbolos personalizados
CustomRatesReplace	Sustituye totalmente la historia de precios del instrumento personalizado en el intervalo temporal indicado, con los datos de la matriz del tipo MqlRates	Símbolos personalizados
CustomRatesUpdate	Añade a la historia del instrumento personalizado las barras ausentes y sustituye las existentes con datos de la matriz del tipo MqlRates	Símbolos personalizados
CustomTicksAdd	Añade a la historia de precios del instrumento personalizado los datos de la matriz del tipo MqlTick. El símbolo personalizado debe ser elegido en la ventana de MarketWatch (Observación del mercado)	Símbolos personalizados

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
CustomTicksDelete	Elimina todos los ticks en el intervalo temporal indicado de la historia de precios del instrumento personalizado	Símbolos personalizados
CustomTicksReplace	Sustituye totalmente la historia de precios del instrumento personalizado en el intervalo temporal indicado, con los datos de la matriz del tipo MqlTick	Símbolos personalizados
CustomBookAdd	Transmite el estado de la profundidad de mercado de un instrumento personalizado	Símbolos personalizados
DatabaseOpen	Abre o crea una base de datos en el archivo indicado	Trabajo con la base de datos
DatabaseClose	Cierra una base de datos	Trabajo con la base de datos
DatabaseImport	Importa a un recuadro los datos de un archivo	Trabajo con la base de datos
DatabaseExport	Exporta un recuadro o el resultado de la ejecución de una solicitud SQL a un archivo CSV	Trabajo con la base de datos
DatabasePrint	Imprime el recuadro o resultado de la ejecución de una solicitud SQL en el diario de expertos	Trabajo con la base de datos
DatabaseTableExists	Comprueba la presencia de un recuadro en la base de datos	Trabajo con la base de datos
DatabaseExecute	Ejecuta una solicitud a la base de datos indicada	Trabajo con la base de datos
DatabasePrepare	Crea el manejador de una solicitud, que después puede ser ejecutada con la ayuda de DatabaseRead()	Trabajo con la base de datos
DatabaseReset	Resetea la solicitud a su estado inicial, igual que tras la llamada de DatabasePrepare()	Trabajo con la base de datos
DatabaseBind	Establece el valor de un parámetro en la solicitud	Trabajo con la base de datos
DatabaseBindArray	Establece una matriz como valor del parámetro.	Trabajo con la base de datos
DatabaseRead	Ejecuta el paso a la siguiente entrada en el resultado de la	Trabajo con la base de datos

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
	solicitud	
DatabaseReadBind	Pasa a la siguiente entrada y lee los datos en la estructura de esta	Trabajo con la base de datos
DatabaseFinalize	Elimina la solicitud creada en DatabasePrepare()	Trabajo con la base de datos
DatabaseTransactionBegin	Comienza la ejecución de una transacción	Trabajo con la base de datos
DatabaseTransactionCommit	Finaliza la ejecución de una transacción	Trabajo con la base de datos
DatabaseTransactionRollback	Ejecuta el retroceso de una transacción	Trabajo con la base de datos
DatabaseColumnsCount	Obtiene el número de campos en una solicitud	Trabajo con la base de datos
DatabaseColumnName	Obtiene el nombre de un campo según el número	Trabajo con la base de datos
DatabaseColumnType	Obtiene el tipo de un campo según el número	Trabajo con la base de datos
DatabaseColumnSize	Obtiene el tamaño del campo en bytes	Trabajo con la base de datos
DatabaseColumnText	Obtiene de la entrada actual el valor del campo en forma de línea	Trabajo con la base de datos
DatabaseColumnInteger	Obtiene de la entrada actual un valor del tipo int	Trabajo con la base de datos
DatabaseColumnLong	Obtiene de la entrada actual un valor del tipo long	Trabajo con la base de datos
DatabaseColumnDouble	Obtiene de la entrada actual un valor del tipo double	Trabajo con la base de datos
DatabaseColumnBlob	Obtiene de la entrada actual el valor del campo en forma de matriz	Trabajo con la base de datos
DebugBreak	Punto programado de interrupción en depuración	Funciones comunes
Digits	Devuelve la cantidad de dígitos decimales después del punto que determina la precisión de medición del precio del símbolo del gráfico corriente	Comprobación de estado

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
DoubleToString	Conversión del valor numérico a una cadena de caracteres con una precisión especificada	Conversión de datos
DXContextCreate	Crea un contexto gráfico para dibujar frames del tamaño indicado	Trabajo con DirectX
DXContextSetSize	Modifica el tamaño del frame de un contexto gráfico creado en DXContextCreate()	Trabajo con DirectX
DXContextSetSize	Obtiene el tamaño del frame de un contexto gráfico creado en DXContextCreate()	Trabajo con DirectX
DXContextClearColors	Establece para el búfer de dibujado todos los píxeles en el color indicado	Trabajo con DirectX
DXContextClearDepth	Limpia el búfer de profundidad	Trabajo con DirectX
DXContextGetColors	Obtiene del contexto gráfico la imagen con el tamaño y desplazamiento indicados	Trabajo con DirectX
DXContextGetDepth	Obtiene el búfer de profundidad del frame dibujado	Trabajo con DirectX
DXBufferCreate	Crea un búfer del tipo indicado basado en una matriz de datos	Trabajo con DirectX
DXTextureCreate	Crea una textura en 2 dimensiones a partir de un rectángulo del tamaño indicado, recortado de la imagen transmitida	Trabajo con DirectX
DXInputCreate	Crea los parámetros de entrada del sombreador	Trabajo con DirectX
DXInputSet	Establece los parámetros de entrada del sombreador	Trabajo con DirectX
DXShaderCreate	Crea un sombreador del tipo indicado	Trabajo con DirectX
DXShaderSetLayout	Establece una marca de vértice para el sombreador de vértices	Trabajo con DirectX
DXShaderInputsSet	Establece los parámetros de entrada del sombreador	Trabajo con DirectX

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
DXShaderTexturesSet	Establece las texturas para el sombreador	Trabajo con DirectX
DXDraw	Dibuja los vértices del búfer de vértices establecido en DXBufferSet()	Trabajo con DirectX
DXDrawIndexed	Dibuja las primitivas gráficas descritas por el búfer de índices de DXBufferSet()	Trabajo con DirectX
DXPrimiveTopologySet	Establece el tipo de primitivas para el dibujado con la ayuda de DXDrawIndexed()	Trabajo con DirectX
DXBufferSet	Establece el búfer para el dibujado actual	Trabajo con DirectX
DXShaderSet	Establece el sombreador para el dibujado	Trabajo con DirectX
DXHandleType	Retorna el tipo de manejador	Trabajo con DirectX
DXRelease	Libera el manejador	Trabajo con DirectX
EnumToString	Conversión del valor de una enumeración de cualquier tipo a una cadena de caracteres	Conversión de datos
EventChartCustom	Genera los eventos personalizados para el gráfico especificado	Trabajo con eventos
EventKillTimer	Detiene la generación de eventos en el gráfico actual según el temporizador	Trabajo con eventos
EventSetMillisecondTimer	Arranca el generador de eventos del temporizador de alta precisión con el período inferior a 1 segundo para el gráfico actual	Trabajo con eventos
EventSetTimer	Arranca el generador de eventos de temporizador con la periodicidad especificada para el gráfico actual	Trabajo con eventos
exp	Devuelve el exponente de un número	Funciones matemáticas
ExpertRemove	Detiene el trabajo del Asesor Experto y lo descarga del gráfico	Funciones comunes
fabs	Devuelve el valor absoluto (modular) de un número que se le	Funciones matemáticas

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
	ha pasado	
FileClose	Cierra un archivo previamente abierto	Operaciones con archivos
FileCopy	Copia el archivo original de una carpeta local o compartida a otro archivo	Operaciones con archivos
FileDelete	Elimina un archivo especificado	Operaciones con archivos
FileFindClose	Cierra el manejador de búsqueda	Operaciones con archivos
FileFindFirst	Empieza la búsqueda de los archivos en el directorio correspondiente de acuerdo con el filtro especificado	Operaciones con archivos
FileFindNext	Sigue con la búsqueda empezada por la función FileFindFirst()	Operaciones con archivos
FileFlush	Guarda en el disco todos los datos que se han quedado en el buffer de entrada/salida	Operaciones con archivos
FileGetInteger	Obtiene una propiedad del número entero del archivo	Operaciones con archivos
FileIsEnding	Determina el final de un archivo en el proceso de lectura	Operaciones con archivos
FileIsExist	Comprueba la existencia de un archivo	Operaciones con archivos
FileIsLineEnding	Determina el fin de una línea en un archivo de texto en el proceso de lectura	Operaciones con archivos
FileMove	Mueve o renombra un archivo	Operaciones con archivos
FileOpen	Abre un archivo con el nombre y banderas especificados	Operaciones con archivos
FileReadArray	Lee los arrays de cualquier tipo, salvo los arrays literales (string) (puede ser un array de estructuras que no contienen las cadenas ni arrays dinámicos) de un archivo binario desde la posición actual del puntero de archivos	Operaciones con archivos
FileReadBool	Lee de un archivo del tipo CSV una cadena desde la posición actual hasta el separador (o hasta	Operaciones con archivos

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
	el final de la línea de texto) y convierte la cadena leída al valor del tipo bool	
FileReadDatetime	Lee de un archivo del tipo CSV una cadena de uno de los formatos: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" o "HH:MI:SS" - y la convierte al valor del tipo datetime	Operaciones con archivos
FileReadDouble	Lee un número de doble precisión con punto flotante (double) de un archivo binario desde la posición actual del puntero de archivos	Operaciones con archivos
FileReadFloat	Lee desde la posición actual del puntero de archivos valor del tipo float	Operaciones con archivos
FileReadInteger	Lee de un archivo binario valor del tipo int, short o char dependiendo de la longitud indicada en bytes	Operaciones con archivos
FileReadLong	Lee desde la posición actual del puntero de archivos valor del tipo long	Operaciones con archivos
FileReadNumber	Lee de un archivo del tipo CSV una cadena desde la posición actual hasta el separador (o hasta el final de la línea de texto) y convierte la cadena leída al valor del tipo double	Operaciones con archivos
FileReadString	Lee de un archivo una cadena desde la posición actual del puntero de archivos	Operaciones con archivos
FileReadStruct	Lee de un archivo binario el contenido en una estructura que ha sido pasada como un parámetro	Operaciones con archivos
FileSeek	Mueve la posición del puntero de archivos a una cantidad de bytes especificada respecto a la posición indicada	Operaciones con archivos
FileSize	Devuelve el tamaño de un archivo correspondiente abierto	Operaciones con archivos

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
FileTell	Devuelve la posición actual del puntero de archivos de un archivo correspondiente abierto	Operaciones con archivos
FileWrite	Escribe los datos en un archivo del tipo CSV o TXT	Operaciones con archivos
FileWriteArray	Escribe los arrays de cualquier tipo (excepto los arrays string) en un archivo del tipo BIN	Operaciones con archivos
FileWriteDouble	Escribe el valor del parámetro del tipo double desde la posición actual del puntero de archivos en un archivo binario	Operaciones con archivos
FileWriteFloat	Escribe el valor del parámetro del tipo float desde la posición actual del puntero de archivos en un archivo binario	Operaciones con archivos
FileWriteInteger	Escribe el valor del parámetro del tipo int desde la posición actual del puntero de archivos en un archivo binario	Operaciones con archivos
FileWriteLong	Escribe el valor del parámetro del tipo long desde la posición actual del puntero de archivos en un archivo binario	Operaciones con archivos
FileWriteString	Escribe el valor del parámetro del tipo string desde la posición actual del puntero de archivos en un archivo del tipo BIN o TXT	Operaciones con archivos
FileWriteStruct	Escribe el contenido de una estructura pasada como un parámetro en un archivo binario desde la posición actual del puntero de archivos	Operaciones con archivos
floor	Devuelve el valor numérico entero más cercano desde abajo	Funciones matemáticas
fmax	Devuelve el valor máximo de dos valores numéricos	Funciones matemáticas
fmin	Devuelve el valor mínimo de dos valores numéricos	Funciones matemáticas
fmod	Devuelve el resto real de la división de dos números	Funciones matemáticas

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
FolderClean	Elimina todos los archivos en la carpeta especificada	Operaciones con archivos
FolderCreate	Crea un directorio en la carpeta Files (dependiendo del valor common_flag)	Operaciones con archivos
FolderDelete	Elimina un directorio seleccionado. Una carpeta no vacía no puede ser eliminada	Operaciones con archivos
FrameAdd	Añade un frame con datos	Trabajo con resultados de optimización
FrameFilter	Establece el filtro de lectura de frames y mueve el puntero al inicio	Trabajo con resultados de optimización
FrameFirst	Mueve el puntero de lectura de frames al inicio y reinicia el filtro establecido antes	Trabajo con resultados de optimización
FrameInputs	Recibe los parámetros input sobre los que está formado el frame	Trabajo con resultados de optimización
FrameNext	Lee el frame y mueve el puntero al siguiente	Trabajo con resultados de optimización
GetLastError	Devuelve el valor del último error	Comprobación de estado
GetPointer	Devuelve el puntero a objeto	Funciones comunes
GetTickCount	Devuelve la cantidad de millisegundos pasados desde el momento del arranque del sistema	Funciones comunes
GlobalVariableCheck	Comprueba la existencia de una variable global con el nombre especificado	Variables globales del terminal de cliente
GlobalVariableDel	Elimina una variable global	Variables globales del terminal de cliente
GlobalVariableGet	Solicita el valor de una variable global	Variables globales del terminal de cliente
GlobalVariableName	Devuelve el nombre de una variable global según su número ordinal en la lista de variables globales	Variables globales del terminal de cliente
GlobalVariablesDeleteAll	Elimina variables globales con el prefijo especificado en su nombre	Variables globales del terminal de cliente

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
GlobalVariableSet	Establece el nuevo valor para una variable global	Variables globales del terminal de cliente
GlobalVariableSetOnCondition	Establece el nuevo valor de una variable global ya existente según una condición	Variables globales del terminal de cliente
GlobalVariablesFlush	Guarda por vía forzada el contenido de todas las variables globales en el disco	Variables globales del terminal de cliente
GlobalVariablesTotal	Devuelve la cantidad total de variables globales	Variables globales del terminal de cliente
GlobalVariableTemp	Establece el nuevo valor para una variable global que existe sólo durante esta sesión del terminal	Variables globales del terminal de cliente
GlobalVariableTime	Devuelve la hora del último acceso a una variable global	Variables globales del terminal de cliente
HistoryDealGetDouble	Devuelve la propiedad solicitada de una transacción en el historial (double)	Funciones comerciales
HistoryDealGetInteger	Devuelve la propiedad solicitada de una transacción en el historial (datetime o int)	Funciones comerciales
HistoryDealGetString	Devuelve la propiedad solicitada de una transacción en el historial (string)	Funciones comerciales
HistoryDealGetTicket	Elige una transacción a procesar y devuelve el ticket de transacción en el historial	Funciones comerciales
HistoryDealSelect	Elige en el historial una transacción para dirigirse a ella en el futuro mediante las funciones correspondientes	Funciones comerciales
HistoryDealsTotal	Devuelve el número de transacciones en el historial	Funciones comerciales
HistoryOrderGetDouble	Devuelve la propiedad solicitada de una orden en el historial (double)	Funciones comerciales
HistoryOrderGetInteger	Devuelve la propiedad solicitada de una orden en el historial (datetime o int)	Funciones comerciales

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
HistoryOrderGetString	Devuelve la propiedad solicitada de una orden en el historial (string)	Funciones comerciales
HistoryOrderGetTicket	Devuelve el ticket de una orden correspondiente en el historial	Funciones comerciales
HistoryOrderSelect	Elige en el historial una orden para el futuro trabajo con ella	Funciones comerciales
HistoryOrdersTotal	Devuelve el número de órdenes en el historial	Funciones comerciales
HistorySelect	Solicita el historial de transacciones y órdenes del período especificado de la hora del servidor	Funciones comerciales
HistorySelectByPosition	Solicita el historial de transacciones y órdenes con el identificador de posición especificado	Funciones comerciales
iBars	Retorna el número de barras en la historia según el símbolo y el periodo correspondientes	Acceso a las series temporales y a los datos de indicadores
iBarShift	Busca la barra según la hora y fecha. La función retorna el índice de la barra en el que entra la hora y fecha especificada	Acceso a las series temporales y a los datos de indicadores
iClose	Retorna el valor del precio de cierre de la barra (indicada por el parámetro shift) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores
iHigh	Retorna el valor del precio mínimo de la barra (indicada por el parámetro shift) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores
iHighest	Retorna el índice del mayor valor encontrado (desplazamiento relativo a la barra actual) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores
iLow	Retorna el valor del precio mínimo de la barra (indicada por el parámetro shift) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
iLowest	Retorna el índice del menor valor encontrado (desplazamiento relativo a la barra actual) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores
iOpen	Retorna el valor del precio de apertura de la barra (indicada por el parámetro shift) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores
iTime	Retorna el valor del tiempo de apertura de la barra (indicada por el parámetro shift) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores
iTickVolume	Retorna el valor del volumen de ticks de la barra (indicada por el parámetro shift) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores
iRealVolume	Retorna el valor del volumen real de la barra (indicada por el parámetro shift) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores
iVolume	Retorna el valor del volumen de ticks de la barra (indicada por el parámetro shift) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores
iSpread	Retorna el valor del spread de la barra (indicada por el parámetro shift) del gráfico correspondiente	Acceso a las series temporales y a los datos de indicadores
iAD	Accumulation/Distribution	Indicadores técnicos
iADX	Average Directional Index	Indicadores técnicos
iADXWilder	Average Directional Index by Welles Wilder	Indicadores técnicos
iAlligator	Alligator	Indicadores técnicos
iAMA	Adaptive Moving Average	Indicadores técnicos
iAO	Awesome Oscillator	Indicadores técnicos
iATR	Average True Range	Indicadores técnicos
iBands	Bollinger Bands®	Indicadores técnicos
iBearsPower	Bears Power	Indicadores técnicos
iBullsPower	Bulls Power	Indicadores técnicos

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
iBWMFI	Market Facilitation Index by Bill Williams	Indicadores técnicos
iCCI	Commodity Channel Index	Indicadores técnicos
iChaikin	Chaikin Oscillator	Indicadores técnicos
iCustom	indicador personalizado	Indicadores técnicos
iDEMA	Double Exponential Moving Average	Indicadores técnicos
iDeMarker	DeMarker	Indicadores técnicos
iEnvelopes	Envelopes	Indicadores técnicos
iForce	Force Index	Indicadores técnicos
iFractals	Fractals	Indicadores técnicos
iFrAMA	Fractal Adaptive Moving Average	Indicadores técnicos
iGator	Gator Oscillator	Indicadores técnicos
iIchimoku	Ichimoku Kinko Hyo	Indicadores técnicos
iMA	Moving Average	Indicadores técnicos
iMACD	Moving Averages Convergence-Divergence	Indicadores técnicos
iMFI	Money Flow Index	Indicadores técnicos
iMomentum	Momentum	Indicadores técnicos
IndicatorCreate	Devuelve el manejador (handle) del indicador técnico especificado que ha sido creado a base del array de parámetros del tipo MqlParam	Acceso a las series temporales y a los datos de indicadores
IndicatorParameters	Devuelve para el manejador especificado el número de los parámetros de entrada del indicador, así como los propios valores y el tipo de parámetros	Acceso a las series temporales y a los datos de indicadores
IndicatorRelease	Elimina el manejador (handle) del indicador y libera la parte calculadora del indicador si nadie la está usando	Acceso a las series temporales y a los datos de indicadores
IndicatorSetDouble	Establece el valor de una propiedad del indicador que tiene el tipo double	Indicadores personalizados

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
IndicatorSetInteger	Establece el valor de una propiedad del indicador que tiene el tipo int	Indicadores personalizados
IndicatorSetString	Establece el valor de una propiedad del indicador que tiene el tipo string	Indicadores personalizados
IntegerToString	Conversión del valor del tipo entero a una cadena de longitud definida	Conversión de datos
iOBV	On Balance Volume	Indicadores técnicos
iOsMA	Moving Average of Oscillator (MACD histogram)	Indicadores técnicos
iRSI	Relative Strength Index	Indicadores técnicos
iRVI	Relative Vigor Index	Indicadores técnicos
iSAR	Parabolic Stop And Reverse System	Indicadores técnicos
IsStopped	Devuelve true, si se ha recibido el comando de terminar la ejecución de un programa mql5	Comprobación de estado
iStdDev	Standard Deviation	Indicadores técnicos
iStochastic	Stochastic Oscillator	Indicadores técnicos
iTEMA	Triple Exponential Moving Average	Indicadores técnicos
iTriX	Triple Exponential Moving Averages Oscillator	Indicadores técnicos
iVIDyA	Variable Index Dynamic Average	Indicadores técnicos
iVolumes	Volumes	Indicadores técnicos
iWPR	Williams' Percent Range	Indicadores técnicos
log	Devuelve un logaritmo neperiano (natural)	Funciones matemáticas
log10	Devuelve el logaritmo de un número en base 10	Funciones matemáticas
MarketBookAdd	Proporciona la apertura de la profundidad de mercado (Depth Market) para el símbolo indicado, además se encarga de la suscripción a las notificaciones	Obtención de información de mercado

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
	acerca del cambio de esta profundidad de mercado	
MarketBookGet	Devuelve un array de estructuras del tipo MqlBookInfo que contiene datos de la profundidad de mercado del símbolo especificado	Obtención de información de mercado
MarketBookRelease	Proporciona el cierre de la profundidad de mercado (Depth Market) para el símbolo indicado, además da de baja la suscripción a las notificaciones acerca del cambio de esta profundidad de mercado	Obtención de información de mercado
MathAbs	Devuelve el valor absoluto (modular) de un número que se le ha pasado	Funciones matemáticas
MathArccos	Devuelve el valor de arcocoseno de (x) en radianes	Funciones matemáticas
MathArcsin	Devuelve el valor de arcseno de (x) en radianes	Funciones matemáticas
MathArctan	Devuelve el valor de arcotangete de (x) en radianes	Funciones matemáticas
MathCeil	Devuelve el valor numérico entero más cercano desde arriba	Funciones matemáticas
MathCos	Devuelve el coseno de un número	Funciones matemáticas
MathExp	Devuelve el exponente de un número	Funciones matemáticas
MathFloor	Devuelve el valor numérico entero más cercano desde abajo	Funciones matemáticas
MathIsValidNumber	Verifica la correctitud de un número real	Funciones matemáticas
MathLog	Devuelve un logaritmo neperiano (natural)	Funciones matemáticas
MathLog10	Devuelve el logaritmo de un número en base 10	Funciones matemáticas
MathMax	Devuelve el valor máximo de dos valores numéricos	Funciones matemáticas
MathMin	Devuelve el valor mínimo de dos valores numéricos	Funciones matemáticas

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
MathMod	Devuelve el resto real de la división de dos números	Funciones matemáticas
MathPow	Eleva la base a la potencia indicada	Funciones matemáticas
MathRand	Devuelve un número pseudoaleatorio en el rango de 0 a 32767	Funciones matemáticas
MathRound	Redondea un número hasta el entero más cercano	Funciones matemáticas
MathSin	Devuelve el seno de un número	Funciones matemáticas
MathSqrt	Devuelve una raíz cuadrada	Funciones matemáticas
MathSrand	Define el estado inicial del generador pseudoaleatorio de números enteros	Funciones matemáticas
MathTan	Devuelve la tangete de un número	Funciones matemáticas
MessageBox	Crea y visualiza la ventana de mensajes, además de gestionarlo	Funciones comunes
MQLInfoInteger	Devuelve un valor del tipo entero de una propiedad correspondiente de un programa mql5 en funcionamiento	Comprobación de estado
MQLInfoString	Devuelve un valor del tipo string de una propiedad correspondiente de un programa mql5 en funcionamiento	Comprobación de estado
MT5Initialize	Establece una conexión con el terminal MetaTrader 5	MetaTrader para Python
MT5Shutdown	Cierra una conexión anteriormente establecida con el terminal MetaTrader 5	MetaTrader para Python
MT5TerminalInfo	Obtiene el estado y los parámetros del terminal MetaTrader 5 conectado	MetaTrader para Python
MT5Version	Retorna la versión del terminal MetaTrader 5	MetaTrader para Python
MT5CopyRatesFrom	Obtiene las barras del terminal MetaTrader 5, a partir de la fecha indicada	MetaTrader para Python

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
MT5CopyRatesFromPos	Obtiene las barras del terminal MetaTrader 5, a partir del índice establecido	MetaTrader para Python
MT5CopyRatesRange	Obtiene las barras en el intervalo de fechas indicado del terminal MetaTrader 5	MetaTrader para Python
MT5CopyTicksFrom	Obtiene los ticks del terminal MetaTrader 5, a partir de la fecha indicada	MetaTrader para Python
MT5CopyTicksRange	Obtiene los ticks en el intervalo de fechas indicado del terminal MetaTrader 5	MetaTrader para Python
NormalizeDouble	Redondeo de un número con punto flotante hasta una precisión especificada	Conversión de datos
ObjectCreate	Crea un objeto del tipo especificado en un gráfico especificado	Objetos gráficos
ObjectDelete	Elimina el objeto con el nombre especificado del gráfico especificado (subventana del gráfico especificada)	Objetos gráficos
ObjectFind	Busca por el nombre un objeto con el identificador especificado	Objetos gráficos
ObjectGetDouble	Devuelve el valor del tipo double de la propiedad correspondiente de un objeto	Objetos gráficos
ObjectGetInteger	Devuelve el valor de números enteros de la propiedad correspondiente de un objeto	Objetos gráficos
ObjectGetString	Devuelve el valor del tipo string de la propiedad correspondiente de un objeto	Objetos gráficos
ObjectGetTimeByValue	Devuelve el valor de hora para el valor especificado del precio de un objeto	Objetos gráficos
ObjectGetValueByTime	Devuelve el valor de precio de un objeto para la hora especificada	Objetos gráficos
ObjectMove	Cambia las coordinadas del punto de anclaje de un objeto	Objetos gráficos

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
ObjectName	Devuelve el nombre de un objeto del tipo correspondiente en el gráfico especificado (subventana del gráfico especificada)	Objetos gráficos
ObjectsDeleteAll	Elimina todos los objetos del tipo especificado del gráfico especificado (subventana del gráfico especificada)	Objetos gráficos
ObjectSetDouble	Establece el valor de propiedad correspondiente de un objeto	Objetos gráficos
ObjectSetInteger	Establece el valor de propiedad correspondiente de un objeto	Objetos gráficos
ObjectSetString	Establece el valor de propiedad correspondiente de un objeto	Objetos gráficos
ObjectsTotal	Devuelve el número de objetos del tipo especificado en el gráfico especificado (subventana del gráfico especificada)	Objetos gráficos
OnStart	Se llama en un script al suceder el evento Start para la ejecución de las acciones implementadas en el script	Procesamiento de eventos
OnInit	Se llama en los indicadores y expertos al suceder el evento Init para inicializar un programa MQL5 en marcha	Procesamiento de eventos
OnDeinit	Se llama en los indicadores y expertos al suceder el evento Deinit para desinicializar un programa MQL5 en marcha	Procesamiento de eventos
OnTick	Se llama en los expertos al suceder el evento NewTick para el procesamiento de una nueva cotización	Procesamiento de eventos
OnCalculate	Se llama en los indicadores al suceder el evento Calculate para el procesamiento del cambio en los datos de precio	Procesamiento de eventos
OnTimer	Se llama en los indicadores y expertos al suceder el evento periódico Timer, generado por el	Procesamiento de eventos

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
	terminal con el intervalo de precio establecido	
OnTrade	Se llama en los expertos al suceder el evento Trade, generado al finalizar una operación comercial en el servidor comercial	Procesamiento de eventos
OnTradeTransaction	Se llama en los expertos al suceder el evento TradeTransaction para el procesamiento de los resultados de la ejecución de una solicitud comercial	Procesamiento de eventos
OnBookEvent	Se llama en los expertos al suceder el evento BookEvent para el procesamiento de los cambios en la profundidad de mercado	Procesamiento de eventos
OnChartEvent	Se llama en los indicadores y expertos al suceder el evento ChartEvent para el procesamiento de los cambios del gráfico provocados por las acciones del usario o el funcionamiento de los programas MQL5	Procesamiento de eventos
OnTester	Se llama en los expertos al suceder el evento Tester para el procesamiento de las acciones necesarias al finalizar la simulación	Procesamiento de eventos
OnTesterInit	Se llama en los expertos al suceder el evento TesterInit para el procesamiento de las acciones necesarias antes de comenzar la optimización	Procesamiento de eventos
OnTesterDeinit	Se llama en los expertos al suceder el evento TesterDeinit para la ejecución de las acciones necesarias al finalizar la optimización del experto	Procesamiento de eventos
OnTesterPass	Se llama en los expertos al suceder el evento TesterPass para el procesamiento de un nuevo frame de datos durante la optimización del experto	Procesamiento de eventos

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
OrderCalcMargin	Calcula el margen requerido para el tipo de orden especificado en la moneda de depósito de la cuental	Funciones comerciales
OrderCalcProfit	Calcula el beneficio basado en los parámetros pasados en la moneda de depósito de la cuenta	Funciones comerciales
OrderCheck	Comprueba si la cuenta dispone de fondos suficientes para ejecutar la operación comercial requerida	Funciones comerciales
OrderGetDouble	Devuelve la propiedad solicitada de una orden (double)	Funciones comerciales
OrderGetInteger	Devuelve la propiedad solicitada de una orden (datetime o int)	Funciones comerciales
OrderGetString	Devuelve la propiedad solicitada de una orden (string)	Funciones comerciales
OrderGetTicket	Devuelve el ticket de una orden correspondiente	Funciones comerciales
OrderSelect	Elige una orden para el futuro trabajo con ella	Funciones comerciales
OrderSend	Comprueba si hay suficientes fondos para ejecutar la operación comercial especificada.	Funciones comerciales
OrderSendAsync	Envía de modo asincrónico las solicitudes comerciales sin esperar la respuesta por parte del servidor de trading	Funciones comerciales
OrdersTotal	Devuelve el número de órdenes	Funciones comerciales
ParameterGetRange	Recibe para la variable input la información sobre la banda de valores y el paso de cambios durante la optimización del EA en el Probador de Estrategias	Trabajo con resultados de optimización
ParameterSetRange	Establece las reglas del uso de la variable input durante la optimización del EA en el Probador de Estrategias: valor, paso de cambio, valor inicial y final	Trabajo con resultados de optimización
Period	Devuelve el período de tiempo del gráfico corriente	Comprobación de estado

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
PeriodSeconds	Devuelve la cantidad de segundos en el período	Funciones comunes
PlaySound	Reproduce un archivo de audio	Funciones comunes
PlotIndexGetInteger	Devuelve el valor de propiedad de línea del indicador que tiene el tipo entero	Indicadores personalizados
PlotIndexSetDouble	Establece el valor de propiedad de línea del indicador que tiene el tipo double	Indicadores personalizados
PlotIndexSetInteger	Establece el valor de propiedad de línea del indicador que tiene el tipo int	Indicadores personalizados
PlotIndexSetString	Establece el valor de propiedad de línea del indicador que tiene el tipo string	Indicadores personalizados
Point	Devuelve el tamaño del punto del instrumento actual en divisa de cotización	Comprobación de estado
PositionGetDouble	Devuelve la propiedad solicitada de una posición abierta (double)	Funciones comerciales
PositionGetInteger	Devuelve la propiedad solicitada de una posición abierta (datetime o int)	Funciones comerciales
PositionGetString	Devuelve la propiedad solicitada de una posición abierta (string)	Funciones comerciales
PositionGetSymbol	Devuelve el símbolo de una posición correspondiente abierta	Funciones comerciales
PositionSelect	Elige una posición abierta para el futuro trabajo con ella	Funciones comerciales
PositionsTotal	Devuelve el número de posiciones abiertas	Funciones comerciales
pow	Eleva la base a la potencia indicada	Funciones matemáticas
Print	Visualiza un mensaje en el registro	Funciones comunes
PrintFormat	Formatea e imprime juego de símbolos y valores en un registro histórico de acuerdo con el formato establecido	Funciones comunes

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
rand	Devuelve un número pseudoaleatorio en el rango de 0 a 32767	Funciones matemáticas
ResetLastError	Pone el valor de variable predefinida _LastError a cero	Funciones comunes
ResourceCreate	Esta función crea un recurso de imagen a base de un conjunto de datos	Funciones comunes
ResourceFree	Elimina el recurso creado dinámicamente (libera la memoria ocupada por el recurso)	Funciones comunes
ResourceReadImage	Lee los datos del recurso gráfico creado con la función ResourceCreate() o guardado en el archivo EX5 tras la compilación	Funciones comunes
ResourceSave	Guarda el recurso en el archivo especificado	Funciones comunes
round	Redondea un número hasta el entero más cercano	Funciones matemáticas
SendFTP	Envía un archivo a la dirección especificada en la ventana de configuraciones, en la pestaña "Edición"	Funciones comunes
SendMail	Envía una carta electrónica a la dirección especificada en la ventana de configuraciones, en la pestaña "Edición"	Funciones comunes
SendNotification	Esta función envía las notificaciones Push a los terminales móviles cuyos MetaQuotes ID han sido indicados en la pestaña "Notificaciones"	Funciones comunes
SeriesInfoInteger	Devuelve la información sobre el estado de datos históricos	Acceso a las series temporales y a los datos de indicadores
SetIndexBuffer	Enlaza el búfer de indicadores especificado con el array dinámico unidimensional del tipo double	Indicadores personalizados
ShortArrayToString	Copia una parte del array en una cadena	Conversión de datos

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
ShortToString	Conversión del código del símbolo (unicode) a una cadena de un carácter	Conversión de datos
SignalBaseGetDouble	Devuelve el valor de la propiedad del tipo double para la señal seleccionada	Trade Signals
SignalBaseGetInteger	Devuelve el valor de la propiedad del tipo integer para la señal seleccionada	Trade Signals
SignalBaseGetString	Devuelve el valor de la propiedad del tipo string para la señal seleccionada	Trade Signals
SignalBaseSelect	Selecciona la señal de la base de datos de las señales disponibles en el terminal	Trade Signals
SignalBaseTotal	Devuelve el número total de señales disponibles en el terminal	Trade Signals
SignalInfoGetDouble	Devuelve el valor de la propiedad del tipo double de los ajustes del copiado de la señal	Trade Signals
SignalInfoGetInteger	Devuelve el valor de la propiedad del tipo integer de los ajustes del copiado de la señal	Trade Signals
SignalInfoGetString	Devuelve el valor de la propiedad del tipo string de los ajustes del copiado de la señal	Trade Signals
SignalInfoSetDouble	Establece el valor de la propiedad del tipo double en los ajustes del copiado de la señal	Trade Signals
SignalInfoSetInteger	Establece el valor de la propiedad del tipo integer en los ajustes del copiado de la señal	Trade Signals
SignalSubscribe	Realiza la suscripción al copiado de la señal	Trade Signals
SignalUnsubscribe	Da de baja la suscripción al copiado de la señal	Trade Signals
sin	Devuelve el seno de un número	Funciones matemáticas
Sleep	Suspende la ejecución del Asesor Experto en curso o del script por un intervalo determinado	Funciones comunes

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
SocketCreate	Crea un socket con las banderas indicadas y retorna su manejador	Funciones de red
SocketClose	Cierra el socket	Funciones de red
SocketConnect	Ejecuta la conexión al servidor, con control de límite de tiempo	Funciones de red
SocketIsConnected	Comprueba si está conectado el socket en el momento actual	Funciones de red
SocketIsReadable	Obtiene el número de bytes que se pueden leer desde el socket	Funciones de red
SocketIsWritable	Comprueba si es posible registrar datos en el socket en el momento actual	Funciones de red
SocketTimeouts	Establece el límite de tiempo para obtener y enviar los datos para el objeto de sistema del socket	Funciones de red
SocketRead	Lee los datos desde el socket	Funciones de red
SocketSend	Registra los datos en el socket	Funciones de red
SocketTlsHandshake	Inicia una conexión TLS (SSL) protegida con el host indicado según el protocolo TLS Handshake	Funciones de red
SocketTlsCertificate	Obtiene los datos sobre el certificado usado para proteger la conexión de red	Funciones de red
SocketTlsRead	Lee los datos de una conexión TLS protegida	Funciones de red
SocketTlsReadAvailable	Lee todos los datos disponibles de una conexión TLS protegida	Funciones de red
SocketTlsSend	Envía los datos a través de una conexión TLS protegida	Funciones de red
sqrt	Devuelve una raíz cuadrada	Funciones matemáticas
srand	Define el estado inicial del generador pseudoaleatorio de números enteros	Funciones matemáticas
StringAdd	Añade una subcadena especificada a una cadena en el lugar	Funciones de cadenas de caracteres
StringBufferLen	Devuelve el tamaño del buffer distribuido para una cadena	Funciones de cadenas de caracteres

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
StringCompare	Compara dos cadenas de caracteres, y devuelve 1 si la primera cadena es más grande que la segunda, devuelve 0 si las cadenas son iguales, y -1 (menos 1) si la primera es menor que la segunda.	Funciones de cadenas de caracteres
StringConcatenate	Forma una cadena con los parámetros pasados	Funciones de cadenas de caracteres
StringFill	Rellena una cadena especificada con símbolos especificados en el lugar	Funciones de cadenas de caracteres
StringFind	Busca una subcadena en una cadena	Funciones de cadenas de caracteres
StringFormat	Conversión de un número a una cadena conforme al formato especificado	Conversión de datos
StringGetCharacter	Devuelve el valor de un símbolo que se encuentra en la posición especificada de una cadena	Funciones de cadenas de caracteres
StringInit	Inicializa una cadena con símbolos especificados y proporciona la longitud especificada de una cadena	Funciones de cadenas de caracteres
StringLen	Devuelve el número de símbolos de una cadena	Funciones de cadenas de caracteres
StringReplace	Reemplaza todas las subcadenas encontradas en una cadena de caracteres por una secuencia de símbolos.	Funciones de cadenas de caracteres
StringSetCharacter	Devuelve copia de una cadena con el valor del símbolo modificado en una posición especificada	Funciones de cadenas de caracteres
StringSplit	Obtiene las subcadenas por un separador establecido desde la cadena especificada y devuelve el número de subcadenas obtenidas	Funciones de cadenas de caracteres
StringSubstr	Extrae una subcadena de una cadena de texto que se empieza desde una posición especificada	Funciones de cadenas de caracteres

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
StringToCharArray	Copia los caracteres de una cadena transformada de Unicode en ANSI en una parte seleccionada de un array del tipo uchar	Conversión de datos
StringToColor	Conversión de una cadena del tipo "R,G,B" o una cadena que contiene nombre de un color al valor del tipo color	Conversión de datos
StringToDouble	Conversión de una cadena que contiene la representación simbólica de un número a un número del tipo double	Conversión de datos
StringToInteger	Conversión de una cadena que contiene la representación simbólica de un número a un número del tipo int	Conversión de datos
StringToLower	Transforma todos los símbolos de una cadena indicada en minúsculas en el lugar	Funciones de cadenas de caracteres
StringToShortArray	Copia caracteres de una cadena en una parte especificada de un array del tipo ushort	Conversión de datos
StringToTime	Conversión de una cadena que contiene la hora y/o la fecha en el formato "yyyy.mm.dd [hh:mi]" al número del tipo datetime	Conversión de datos
StringToUpper	Transforma todos los símbolos de una cadena indicada en mayúsculas en el lugar	Funciones de cadenas de caracteres
StringTrimLeft	Borra los símbolos de salto de línea, espacios y símbolos de tabulación en la parte izquierda de la cadena	Funciones de cadenas de caracteres
StringTrimRight	Borra los símbolos de salto de línea, espacios y símbolos de tabulación en la parte derecha de la cadena	Funciones de cadenas de caracteres
StructToTime	Convierte una variable del tipo de estructura MqlDateTime a un valor del tipo datetime	Fecha y hora
Symbol	Devuelve el nombre de un símbolo del gráfico corriente	Comprobación de estado

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
SymbolInfoDouble	Devuelve valor double del símbolo especificado para la propiedad correspondiente	Obtención de información de mercado
SymbolInfoInteger	Devuelve el valor del tipo entero (long, datetime, int o bool) del símbolo especificado para la propiedad correspondiente	Obtención de información de mercado
SymbolInfoMarginRate	Devuelve los coeficientes del margen dependiendo del tipo y la dirección de la orden	Obtención de información de mercado
SymbolInfoSessionQuote	Permite obtener fecha y hora de apertura y de cierre de la sesión de cotización especificada para el símbolo y el día de la semana especificados.	Obtención de información de mercado
SymbolInfoSessionTrade	Permite obtener fecha y hora de apertura y de cierre de la sesión de compraventa especificada para el símbolo y el día de la semana especificados.	Obtención de información de mercado
SymbolInfoString	Devuelve el valor string del símbolo especificado para la propiedad correspondiente	Obtención de información de mercado
SymbolInfoTick	Devuelve los precios actuales para un símbolo especificado en una variable del tipo MqlTick	Obtención de información de mercado
SymbolIsSynchronized	Comprueba la sincronización de datos para el símbolo especificado en el terminal con los datos en el servidor comercial	Obtención de información de mercado
SymbolName	Devuelve el nombre del símbolo especificado	Obtención de información de mercado
SymbolSelect	Selecciona un símbolo en la ventana MarketWatch o remueve un símbolo de la ventana	Obtención de información de mercado
SymbolsTotal	Devuelve la cantidad de símbolos disponibles (seleccionados en MarketWatch o todos)	Obtención de información de mercado
tan	Devuelve la tangete de un número	Funciones matemáticas

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
TerminalClose	Envía al terminal el comando de finalizar el trabajo	Funciones comunes
TerminalInfoDouble	Devuelve un valor del tipo double de una propiedad correspondiente del ambiente de programa mql5	Comprobación de estado
TerminalInfoInteger	Devuelve un valor del tipo entero de una propiedad correspondiente del ambiente de programa mql5	Comprobación de estado
TerminalInfoString	Devuelve un valor del tipo string de una propiedad correspondiente del ambiente de programa mql5	Comprobación de estado
TesterStatistics	La función devuelve el valor del indicador estadístico especificado que ha sido calculado a base de los resultados de simulación	Funciones comunes
TextGetSize	Devuelve el alto y el ancho de la cadena con los ajustes de la fuente actuales	Objetos gráficos
TextOut	Pasa el texto al array personalizado (búfer) que sirve para crear un recurso gráfico	Objetos gráficos
TextSetFont	Establece la fuente para visualizar el texto a través de los métodos del dibujo (por defecto se usa Arial 20)	Objetos gráficos
TimeCurrent	Devuelve la última hora conocida del servidor (la hora de llegada de la última cotización) en el formato datetime	Fecha y hora
TimeDaylightSavings	Devuelve el indicio de cambio horario verano/invierno	Fecha y hora
TimeGMT	Devuelve la hora GMT en el formato datetime, teniendo en cuenta el cambio de horarios de verano o de invierno, según la hora local del ordenador en el que está funcionando el terminal de cliente	Fecha y hora
TimeGMTOffset	Devuelve la diferencia actual entre la hora GMT y hora local del ordenador en segundos, teniendo	Fecha y hora

© 2000-2025, MetaQuotes Ltd.

Función	Acción	Sección
	en cuenta el cambio horario verano/invierno	
TimeLocal	Devuelve la hora local del ordenador en el formato datetime	Fecha y hora
TimeToString	Conversión del valor que contiene el tiempo en segundos transcurridos desde el 01.01.1970 a la cadena con el formato "yyyy.mm.dd hh:mi"	Conversión de datos
TimeToStruct	Convierte un valor del tipo datetime a una variable del tipo de estructura MqlDateTime	Fecha y hora
TimeTradeServer	Devuelve la hora actual de calculación del servidor comercial	Fecha y hora
UninitializeReason	Devuelve el código de la causa de deinicialización	Comprobación de estado
WebRequest	Envía la solicitud HTTP al servidor especificado	Funciones comunes
ZeroMemory	Reinicializa la variable pasada por referencia. La variable puede ser de cualquier tipo, salvo las clases y estructuras que contienen constructores.	Funciones comunes

© 2000-2025, MetaQuotes Ltd.

Lista de Constantes MQL5

Todas las constantes MQL5 en orden alfabético.

Constante	Descripción	Uso
__DATE__	Fecha de compilación del archivo sin hora (horas, minutos y segundos son iguales a 0)	Print
__DATETIME__	Fecha y hora de compilación del archivo	Print
__FILE__	Nombre del archivo corriente compilado	Print
__FUNCSIG__	Signatura de la función en cuyo cuerpo se encuentra la macro. El mostrar en el log la descripción completa de la función	Print

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	con tipos de parámetros puede ser útil durante la identificación de las funciones sobrecargadas	
__FUNCTION__	Nombre de la función, en cuyo cuerpo está ubicado la macro	Print
__LINE__	Número de cadena en el código fuente, en la que se ubica esta macro	Print
__MQLBUILD__, __MQL5BUILD__	Número build del compilador	Print
__PATH__	Ruta absoluta hacia el archivo actual a compilar	Print

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ACCOUNT_ASSETS	Tamaño actual de fondos en la cuenta	AccountInfoDouble
ACCOUNT_BALANCE	Balance de la cuenta en divisa del depósito	AccountInfoDouble
ACCOUNT_COMMISSION_BLOCKED	Importe actual de comisiones bloqueadas de la cuenta	AccountInfoDouble
ACCOUNT_COMPANY	Nombre de la empresa que atiende la cuenta	AccountInfoString
ACCOUNT_CREDIT	Cuantía de crédito concedido en divisa del depósito	AccountInfoDouble
ACCOUNT_CURRENCY	Divisa de depósito	AccountInfoString
ACCOUNT_EQUITY	Cuantía de fondos propios en la	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	cuenta en divisa del depósito	
ACCOUNT_LEVERAGE	Cuantía de apalancamiento concedido	AccountInfoInteger
ACCOUNT_LIABILITIES	Tamaño actual de obligaciones en la cuenta	AccountInfoDouble
ACCOUNT_LIMIT_ORDERS	El número máximo permitido de las órdenes pendientes activas	AccountInfoInteger
ACCOUNT_LOGIN	Número de cuenta	AccountInfoInteger
ACCOUNT_MARGIN	Cuantía de margen reservada en la cuenta en divisa del depósito	AccountInfoDouble
ACCOUNT_MARGIN_FREE	Cuantía de fondos disponibles en la	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	cuenta en divisa del depósito para la apertura de una posición	
ACCOUNT_MARGIN_INITIAL	Cuantía de fondos reservados en la cuenta para cubrir la garantía de todas las órdenes pendientes	AccountInfoDouble
ACCOUNT_MARGIN_LEVEL	Nivel de margen en la cuenta en por cientos	AccountInfoDouble
ACCOUNT_MARGIN_MAINTENANCE	Cuantía de fondos reservados en la cuenta para cubrir el importe mínimo de todas las posiciones abiertas	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ACCOUNT_MARGIN_SO_CALL	Nivel del margen que requiere el recargo de la cuenta. Dependiendo del ACCOUNT_MARGIN_SO_MODE establecido, éste va en por cientos o en divisa del depósito	AccountInfoDouble
ACCOUNT_MARGIN_SO_MODE	Modo de establecimiento del nivel mínimo permitido de la margen	AccountInfoInteger
ACCOUNT_MARGIN_SO_SO	Nivel del margen; al llegar a este nivel, la posición menos rentable se cierra automáticamente. Dependiendo del ACCOUN	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	T_MARGIN_SO_MODE establecido, éste va en por cientos o en divisa del depósito	
ACCOUNT_NAME	Nombre de cliente	AccountInfoString
ACCOUNT_PROFIT	Cuantía del beneficio actual en la cuenta en divisa del depósito	AccountInfoDouble
ACCOUNT_SERVER	Nombre del servidor comercial	AccountInfoString
ACCOUNT_STOPOUT_MODE_MONEY	Nivel en dinero	AccountInfoInteger
ACCOUNT_STOPOUT_MODE_PERCENT	Nivel en por cientos	AccountInfoInteger
ACCOUNT_TRADE_ALLOWED	Permiso del trading para la cuenta corriente	AccountInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ACCOUNT_TRADE_EXPERT	Permiso del trading para un Asesor Experto	AccountInfoInteger
ACCOUNT_TRADE_MODE	Account trade mode	AccountInfoInteger
ACCOUNT_TRADE_MODE_CONTEST	Cuenta comercial de concurso (Contest)	AccountInfoInteger
ACCOUNT_TRADE_MODE_DEMO	Cuenta comercial de demostración (Demo)	AccountInfoInteger
ACCOUNT_TRADE_MODE_REAL	Cuenta comercial real (Real)	AccountInfoInteger
ALIGN_CENTER	Alineación centrada (sólo para el objeto "Campo de edición")	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_LEFT	Alineación por la izquierda	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_RIGHT	Alineación derecha	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ANCHOR_CENTER	Punto de enlace	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	justo en el centro del objeto	
ANCHOR_LEFT	Punto de enlace a la izquierda en el centro	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_LOWER	Punto de enlace en la esquina inferior izquierda	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_UPPER	Punto de enlace en la esquina superior izquierda	ObjectSetInteger, ObjectGetInteger
ANCHOR_LOWER	Punto de enlace abajo en el centro	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT	Punto de enlace a la derecha en el centro	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_LOWER	Punto de enlace en la esquina inferior derecha	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_UPPER	Punto de enlace	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	en la esquina superior derecha	
ANCHOR_UPPER	Punto de enlace arriba en el centro	ObjectSetInteger, ObjectGetInteger
BASE_LINE	Línea principal	Líneas de indicadores
BOOK_TYPE_BUY	Orden de compra	MqlBookInfo
BOOK_TYPE_BUY_MARKET	Solicitud de compra por el precio de mercado	MqlBookInfo
BOOK_TYPE_SELL	Orden de venta	MqlBookInfo
BOOK_TYPE_SELL_MARKET	Solicitud de venta por el precio de mercado	MqlBookInfo
BORDER_FLAT	Borde plano	ObjectSetInteger, ObjectGetInteger
BORDER_RAISED	Convexo	ObjectSetInteger, ObjectGetInteger
BORDER_SUNKEN	Cóncavo	ObjectSetInteger, ObjectGetInteger
CHAR_MAX	Valor máximo que puede ser representado por	Constantes de tipos numéricos

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	el tipo char	
CHAR_MIN	Valor mínimo que puede ser representado por el tipo char	Constantes de tipos numéricos
CHART_AUTOSCROLL	Modo automático de traspaso al lado derecho del gráfico	ChartSetInteger, ChartGetInteger
CHART_BARS	Representación en forma de barras	ChartSetInteger
CHART_BEGIN	Inicio del gráfico (precios más viejos)	ChartNavigate
CHART_BRING_TO_TOP	Visualización del gráfico por encima de los demás	ChartSetInteger, ChartGetInteger
CHART_CANDLES	Representación en forma de velas	ChartSetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	japonesas	
CHART_COLOR_ASK	Color de la línea Ask-precio	ChartSetInteger, ChartGetInteger
CHART_COLOR_BACKGROUND	Color del fondo del gráfico	ChartSetInteger, ChartGetInteger
CHART_COLOR_BID	Color de la línea Bid-precio	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BEAR	Color del cuerpo de la vela oso	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BULL	Color del cuerpo de la vela toro	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_DOWN	Color de barra a la baja, sombras y bordes del cuerpo de la vela de oso	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_LINE	Color de la línea del gráfico y de las velas japonesas "Doji"	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_UP	Color de barra al alza,	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	sombras y bordes del cuerpo de la vela de toro	
CHART_COLOR_FOREGROUND	Color de ejes, escala y línea OHLC	ChartSetInteger, ChartGetInteger
CHART_COLOR_GRID	Color de rejilla	ChartSetInteger, ChartGetInteger
CHART_COLOR_LAST	Color de la línea de precio de la última transacción realizada (Last)	ChartSetInteger, ChartGetInteger
CHART_COLOR_STOP_LEVEL	Color de los niveles de stop-órdenes (Stop Loss y Take Profit)	ChartSetInteger, ChartGetInteger
CHART_COLOR_VOLUME	Color de volúmenes y niveles de apertura de posiciones	ChartSetInteger, ChartGetInteger
CHART_COMMENT	Texto del	ChartSetString, ChartGetString

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	comentario en el gráfico	
CHART_CURRENT_POS	Posición actual	ChartNavigate
CHART_DRAG_TRADE_LEVELS	Permiso para arrastrar los niveles de trading por el gráfico utilizando el ratón. Por defecto, el modo de arrastre está activado (valor true)	ChartSetInteger, ChartGetInteger
CHART_EVENT_MOUSE_MOVE	Envía una notificación sobre los eventos de desplazamiento y pulsación de botones del ratón (CHART_EVENT_MOUSE_MOVE) a todos los programas mql5	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	presentes en el gráfico	
CHART_EVENT_OBJECT_CREATE	Envía una notificación sobre el evento de creación de nuevo objeto gráfico (CHART_EVENT_OBJECT_CREATE) a todos los programas mql5 presentes en el gráfico	ChartSetInteger, ChartGetInteger
CHART_EVENT_OBJECT_DELETE	Envía una notificación sobre el evento de eliminación de un objeto gráfico (CHART_EVENT_OBJECT_DELETE) a todos los programas mql5	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	presentes en el gráfico	
CHART_FIRST_VISIBLE_BAR	Número de la primera barra visible en el gráfico. Indexación de las barras corresponde a la serie temporal.	ChartSetInteger, ChartGetInteger
CHART_FIXED_MAX	Máximo fijo del gráfico	ChartSetDouble, ChartGetDouble
CHART_FIXED_MIN	Mínimo fijo del gráfico	ChartSetDouble, ChartGetDouble
CHART_FIXED_POSITION	Situación de la posición fija del gráfico desde el lado izquierdo en por cientos. La posición fija del gráfico está marcada con un pequeño triángulo gris sobre el	ChartSetDouble, ChartGetDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	eje horizontal de tiempo. Se muestra sólo si está desactivado el desplazamiento automático hacia la derecha con la llegada de un nuevo tick (ver la propiedad CHART_AUTOSCROLL). Cuando aumentamos o disminuimos el zoom, la barra que se encuentra en la posición fija se queda en el mismo sitio.	
CHART_FOREGROUND	Gráfico de precio	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	en el fondo	
CHART_HEIGHT_IN_PIXELS	El alto del gráfico en píxeles	ChartSetInteger, ChartGetInteger
CHART_IS_OBJECT	Indicio para la identificación del objeto "Gráfico" (OBJ_CHART) - para un objeto gráfico devuelve true. Para el gráfico real devuelve false	ChartSetInteger, ChartGetInteger
CHART_LINE	Representación en forma de una línea trazada por los precios Close	ChartSetInteger
CHART_MODE	Tipo de gráfico (velas, barras o líneas)	ChartSetInteger, ChartGetInteger
CHART_MOUSE_SCROLL	Desplazamiento del gráfico por la	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	horizontal usando el botón izquierdo del ratón. Desplazamiento por la vertical también estará disponible si el valor de cualquiera de las siguientes tres propiedades está establecido en true: CHART_SCALEFIX, CHART_SCALEFIX_11 o CHART_SCALE_PT_PER_BAR	
CHART_POINTS_PER_BAR	Valor de la escala en puntos por barra	ChartSetDouble, ChartGetDouble
CHART_PRICE_MAX	Máximo del gráfico	ChartSetDouble, ChartGetDouble
CHART_PRICE_MIN	Mínimo del	ChartSetDouble, ChartGetDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	gráfico	
CHART_SCALE	Escala	ChartSetInteger, ChartGetInteger
CHART_SCALE_PT_PER_BAR	Modo de especificar la escala en puntos por barra	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX	Modo de escala fija	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX_11	Modo de escala 1:1	ChartSetInteger, ChartGetInteger
CHART_SHIFT	Modo de sangría del gráfico de precio desde el lado derecho	ChartSetInteger, ChartGetInteger
CHART_SHIFT_SIZE	Tamaño de sangría de la barra cero desde el lado derecho en por cientos	ChartSetDouble, ChartGetDouble
CHART_SHOW_ASK_LINE	Muestra del valor Ask con una línea horizont	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	al en el gráfico	
CHART_SHOW_BID_LINE	Muestra del valor Bid con una línea horizontal en el gráfico	ChartSetInteger, ChartGetInteger
CHART_SHOW_DATE_SCALE	Visualiza la escala de tiempo en el gráfico	ChartSetInteger, ChartGetInteger
CHART_SHOW_GRID	Muestra de rejilla en el gráfico	ChartSetInteger, ChartGetInteger
CHART_SHOW_LAST_LINE	Muestra del valor Last con una línea horizontal en el gráfico	ChartSetInteger, ChartGetInteger
CHART_SHOW_OBJECT_DESCR	Descripciones emergentes de objetos gráficos	ChartSetInteger, ChartGetInteger
CHART_SHOW_OHLC	Muestra de valores OHLC en esquina izquierda superior	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
CHART_SHOW_ONE_CLICK	Visualización del panel del trading rápido en el gráfico (opción "Trading con un clic")	ChartSetInteger, ChartGetInteger
CHART_SHOW_PERIOD_SEP	Muestra de separadores verticales entre períodos adyacentes	ChartSetInteger, ChartGetInteger
CHART_SHOW_PRICE_SCALE	Visualiza la escala de precios en el gráfico	ChartSetInteger, ChartGetInteger
CHART_SHOW_TRADE_LEVELS	Visualiza en el gráfico los niveles comerciales (niveles de posiciones abiertas, Stop Loss, Take Profit y ordenes pendientes)	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
CHART_SHOW_VOLUMES	Muestra de volúmenes en el gráfico	ChartSetInteger, ChartGetInteger
CHART_VISIBLE_BARS	Cantidad de barras en el gráfico disponibles para ser mostrados	ChartSetInteger, ChartGetInteger
CHART_VOLUME_HIDE	Volúmenes no se muestran	ChartSetInteger
CHART_VOLUME_REAL	Volúmenes comerciales	ChartSetInteger
CHART_VOLUME_TICK	Volúmenes de tick	ChartSetInteger
CHART_WIDTH_IN_BARS	El ancho del gráfico en barras	ChartSetInteger, ChartGetInteger
CHART_WIDTH_IN_PIXELS	El ancho del gráfico en píxeles	ChartSetInteger, ChartGetInteger
CHART_WINDOW_HANDLE	Manejador (handle) del gráfico (HWND)	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
CHART_WINDOW_IS_VISIBLE	Visibilidad de subventanas	ChartSetInteger, ChartGetInteger
CHART_WINDOW_YDISTANCE	Distancia en píxeles por el eje vertical Y entre el marco superior de la subventana del indicador y el marco superior de la ventana principal del gráfico. En caso del evento del ratón, las coordinadas del cursor se pasan en términos de las coordinadas de la ventana principal del gráfico, mientras que las coordina	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	das de los objetos gráficos en la subventana del indicador se establecen respecto a la esquina superior izquierda de la subventana. El valor es necesario para convertir las coordinadas absolutas del gráfico principal a las coordinadas locales de la subventana para un trabajo correcto con los objetos gráficos cuyas coordinadas se establec	

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	en respecto a la esquina superior izquierda del cuadro de la subventana.	
CHART_WINDOWS_TOTAL	Número total de las ventanas del gráfico, incluyendo las subventanas de indicadores	ChartSetInteger, ChartGetInteger
CHARTEVENT_CHART_CHANGE	Modificación de dimensiones del gráfico o de sus propiedades a través del diálogo de propiedades	OnChartEvent
CHARTEVENT_CLICK	Clic en un gráfico	OnChartEvent
CHARTEVENT_CUSTOM	Número inicial de un evento del	OnChartEvent

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	rango de eventos de usuario	
CHARTEVENT_CUSTOM_LAST	Número final de un evento del rango de eventos de usuario	OnChartEvent
CHARTEVENT_KEYDOWN	Teclazos	OnChartEvent
CHARTEVENT_MOUSE_MOVE	Desplazamiento del ratón y pulsación de los botones del ratón (si para el gráfico está establecida la propiedad CHART_EVENT_MOUSE_MOVE=true)	OnChartEvent
CHARTEVENT_OBJECT_CHANGE	Evento de cambio de propiedades de un objeto gráfico a través	OnChartEvent

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	del diálogo de propiedades	
CHARTEVENT_OBJECT_CLICK	Clic en un objeto gráfico	OnChartEvent
CHARTEVENT_OBJECT_CREATE	Creación de un objeto gráfico (si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_CREATE =true)	OnChartEvent
CHARTEVENT_OBJECT_DELETE	Eliminación de un objeto gráfico (si para el gráfico está establecida la propiedad CHART_EVENT_OBJECT_DELETE= true)	OnChartEvent

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
CHARTEVENT_OBJECT_DRAG	Arrastrar un objeto gráfico	OnChartEvent
CHARTEVENT_OBJECT_ENDEDIT	Fin de edición del texto en el objeto gráfico Edit	OnChartEvent
CHARTS_MAX	La cantidad máxima posible de los gráficos abiertos al mismo tiempo en el terminal	Otras constantes
CHIKOUSPAN_LINE	Línea Chikou Span	Líneas de indicadores
clrAliceBlue	Alice Blue	Colores Web
clrAntiqueWhite	Antique White	Colores Web
clrAqua	Aqua	Colores Web
clrAquamarine	Aquamarine	Colores Web
clrBeige	Beige	Colores Web
clrBisque	Bisque	Colores Web
clrBlack	Black	Colores Web
clrBlanchedAlmond	Blanched Almond	Colores Web
clrBlue	Blue	Colores Web

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
clrBlueViolet	Blue Violet	Colores Web
clrBrown	Brown	Colores Web
clrBurlyWood	Burly Wood	Colores Web
clrCadetBlue	Cadet Blue	Colores Web
clrChartreuse	Chartreuse	Colores Web
clrChocolate	Chocolate	Colores Web
clrCoral	Coral	Colores Web
clrCornflowerBlue	Cornflower Blue	Colores Web
clrCornsilk	Cornsilk	Colores Web
clrCrimson	Crimson	Colores Web
clrDarkBlue	Dark Blue	Colores Web
clrDarkGoldenrod	Dark Goldenrod	Colores Web
clrDarkGray	Dark Gray	Colores Web
clrDarkGreen	Dark Green	Colores Web
clrDarkKhaki	Dark Khaki	Colores Web
clrDarkOliveGreen	Dark Olive Green	Colores Web
clrDarkOrange	Dark Orange	Colores Web
clrDarkOrchid	Dark Orchid	Colores Web
clrDarkSalmon	Dark Salmon	Colores Web

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
clrDarkSeaGreen	Dark Sea Green	Colores Web
clrDarkSlateBlue	Dark Slate Blue	Colores Web
clrDarkSlateGray	Dark Slate Gray	Colores Web
clrDarkTurquoise	Dark Turquoise	Colores Web
clrDarkViolet	Dark Violet	Colores Web
clrDeepPink	Deep Pink	Colores Web
clrDeepSkyBlue	Deep Sky Blue	Colores Web
clrDimGray	Dim Gray	Colores Web
clrDodgerBlue	Dodger Blue	Colores Web
clrFireBrick	Fire Brick	Colores Web
clrForestGreen	Forest Green	Colores Web
clrGainsboro	Gainsboro	Colores Web
clrGold	Gold	Colores Web
clrGoldenrod	Goldenrod	Colores Web
clrGray	Gray	Colores Web
clrGreen	Green	Colores Web
clrGreenYellow	Green Yellow	Colores Web
clrHoneydew	Honeydew	Colores Web
clrHotPink	Hot Pink	Colores Web

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
clrIndianRed	Indian Red	Colores Web
clrIndigo	Indigo	Colores Web
clrIvory	Ivory	Colores Web
clrKhaki	Khaki	Colores Web
clrLavender	Lavender	Colores Web
clrLavenderBlush	Lavender Blush	Colores Web
clrLawnGreen	Lawn Green	Colores Web
clrLemonChiffon	Lemon Chiffon	Colores Web
clrLightBlue	Light Blue	Colores Web
clrLightCoral	Light Coral	Colores Web
clrLightCyan	Light Cyan	Colores Web
clrLightGoldenrod	Light Goldenrod	Colores Web
clrLightGray	Light Gray	Colores Web
clrLightGreen	Light Green	Colores Web
clrLightPink	Light Pink	Colores Web
clrLightSalmon	Light Salmon	Colores Web
clrLightSeaGreen	Light Sea Green	Colores Web
clrLightSkyBlue	Light Sky Blue	Colores Web
clrLightSlateGray	Light Slate	Colores Web

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	Gray	
clrLightSteelBlue	Light Steel Blue	Colores Web
clrLightYellow	Light Yellow	Colores Web
clrLime	Lime	Colores Web
clrLimeGreen	Lime Green	Colores Web
clrLinen	Linen	Colores Web
clrMagenta	Magenta	Colores Web
clrMaroon	Maroon	Colores Web
clrMediumAquamarine	Medium Aquamarine	Colores Web
clrMediumBlue	Medium Blue	Colores Web
clrMediumOrchid	Medium Orchid	Colores Web
clrMediumPurple	Medium Purple	Colores Web
clrMediumSeaGreen	Medium Sea Green	Colores Web
clrMediumSlateBlue	Medium Slate Blue	Colores Web
clrMediumSpringGreen	Medium Spring Green	Colores Web
clrMediumTurquoise	Medium Turquoise	Colores Web
clrMediumVioletRed	Medium Violet Red	Colores Web

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
clrMidnightBlue	Midnight Blue	Colores Web
clrMintCream	Mint Cream	Colores Web
clrMistyRose	Misty Rose	Colores Web
clrMoccasin	Moccasin	Colores Web
clrNavajoWhite	Navajo White	Colores Web
clrNavy	Navy	Colores Web
clrNONE	Ausencia de color	Otras constantes
clrOldLace	Old Lace	Colores Web
clrOlive	Olive	Colores Web
clrOliveDrab	Olive Drab	Colores Web
clrOrange	Orange	Colores Web
clrOrangeRed	Orange Red	Colores Web
clrOrchid	Orchid	Colores Web
clrPaleGoldenrod	Pale Goldenrod	Colores Web
clrPaleGreen	Pale Green	Colores Web
clrPaleTurquoise	Pale Turquoise	Colores Web
clrPaleVioletRed	Pale Violet Red	Colores Web
clrPapayaWhip	Papaya Whip	Colores Web
clrPeachPuff	Peach Puff	Colores Web

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
clrPeru	Peru	Colores Web
clrPink	Pink	Colores Web
clrPlum	Plum	Colores Web
clrPowderBlue	Powder Blue	Colores Web
clrPurple	Purple	Colores Web
clrRed	Red	Colores Web
clrRosyBrown	Rosy Brown	Colores Web
clrRoyalBlue	Royal Blue	Colores Web
clrSaddleBrown	Saddle Brown	Colores Web
clrSalmon	Salmon	Colores Web
clrSandyBrown	Sandy Brown	Colores Web
clrSeaGreen	Sea Green	Colores Web
clrSeashell	Seashell	Colores Web
clrSienna	Sienna	Colores Web
clrSilver	Silver	Colores Web
clrSkyBlue	Sky Blue	Colores Web
clrSlateBlue	Slate Blue	Colores Web
clrSlateGray	Slate Gray	Colores Web
clrSnow	Snow	Colores Web
clrSpringGreen	Spring Green	Colores Web
clrSteelBlue	Steel Blue	Colores Web
clrTan	Tan	Colores Web
clrTeal	Teal	Colores Web

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
clrThistle	Thistle	Colores Web
clrTomato	Tomato	Colores Web
clrTurquoise	Turquoise	Colores Web
clrViolet	Violet	Colores Web
clrWheat	Wheat	Colores Web
clrWhite	White	Colores Web
clrWhiteSmoke	White Smoke	Colores Web
clrYellow	Yellow	Colores Web
clrYellowGreen	Yellow Green	Colores Web
CORNER_LEFT_LOWER	Centro de coordinadas se encuentra en la esquina inferior izquierda del gráfico	ObjectSetInteger, ObjectGetInteger
CORNER_LEFT_UPPER	Centro de coordinadas se encuentra en la esquina superior izquierda del gráfico	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_LOWER	Centro de coordinadas se encuentra en la esquina	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	inferior derecha del gráfico	
CORNER_RIGHT_UPPER	Centro de coordinadas se encuentra en la esquina superior derecha del gráfico	ObjectSetInteger, ObjectGetInteger
CP_ACP	Página de código actual de codificación ANSI en el sistema operativo Windows	CharArrayToString, StringToCharArray, FileOpen
CP_MACCP	Página de código actual Macintosh. Nota: Este valor suelen usar en los códigos de programas creados anteriormente, y	CharArrayToString, StringToCharArray, FileOpen

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	actualmente no hay necesidad de usarlo porque los modernos ordenadores Macintosh utilizan la codificación Unicode.	
CP_OEMCP	Página de código actual OEM.	CharArrayToString, StringToCharArray, FileOpen
CP_SYMBOL	Página de código Symbol	CharArrayToString, StringToCharArray, FileOpen
CP_THREAD_ACP	Codificación Windows ANSI para el hilo de ejecución corriente.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF7	Página de código UTF-7.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF8	Página de	CharArrayToString, StringToCharArray, FileOpen

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	código UTF-8.	
CRYPT_AES128	Cifrado AES con la clave de 128 bits (16 bytes)	CryptEncode, CryptDecode
CRYPT_AES256	Cifrado AES con la clave de 256 bits (32 bytes)	CryptEncode, CryptDecode
CRYPT_ARCH_ZIP	Compresión ZIP	CryptEncode, CryptDecode
CRYPT_BASE64	Cifrado BASE64 (recodificación)	CryptEncode, CryptDecode
CRYPT_DES	Cifrado DES con la clave de 56 bits (7 bytes)	CryptEncode, CryptDecode
CRYPT_HASH_MD5	Cálculo HASH MD5	CryptEncode, CryptDecode
CRYPT_HASH_SHA1	Cálculo HASH SHA1	CryptEncode, CryptDecode
CRYPT_HASH_SHA256	Cálculo HASH SHA256	CryptEncode, CryptDecode
DBL_DIG	Número de dígitos decimales significativos	Constantes de tipos numéricos

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
DBL_EPSILON	Valor mínimo que satisface la condición: 1.0+DBL_EPSILON != 1.0	Constantes de tipos numéricos
DBL_MANT_DIG	Cantidad de bits en la mantisa	Constantes de tipos numéricos
DBL_MAX	Valor máximo que puede ser representado por el tipo double	Constantes de tipos numéricos
DBL_MAX_10_EXP	Valor decimal máximo del grado de exponente	Constantes de tipos numéricos
DBL_MAX_EXP	Valor binario máximo del grado de exponente	Constantes de tipos numéricos
DBL_MIN	Valor mínimo positivo que puede ser representado por	Constantes de tipos numéricos

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	el tipo double	
DBL_MIN_10_EXP	Valor decimal máximo del grado de exponente	Constantes de tipos numéricos
DBL_MIN_EXP	Valor binario mínimo del grado de exponente	Constantes de tipos numéricos
DEAL_COMMENT	Comentarios de transacción	HistoryDealGetString
DEAL_COMMISSION	Comisión de transacción	HistoryDealGetDouble
DEAL_ENTRY	Dirección de transacción - entra en el mercado, sale del mercado o da la vuelta	HistoryDealGetInteger
DEAL_ENTRY_IN	Entrada en el mercado	HistoryDealGetInteger
DEAL_ENTRY_INOUT	Vuelta	HistoryDealGetInteger
DEAL_ENTRY_OUT	Salida del mercado	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
DEAL_MAGIC	Magic number para la transacción (véase ORDER_MAGIC)	HistoryDealGetInteger
DEAL_ORDER	Orden a base de la cual la transacción ha sido ejecutada	HistoryDealGetInteger
DEAL_POSITION_ID	Identificador de posición, en la apertura, modificación o cierre de la cual participa esta transacción. Cada posición tiene su único identificador que se asigna a todas las transacciones realizadas con el instrumento durante	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	toda la vida de la posición.	
DEAL_PRICE	Precio de transacción	HistoryDealGetDouble
DEAL_PROFIT	Beneficio de transacción	HistoryDealGetDouble
DEAL_SWAP	Swap acumulado con el cierre	HistoryDealGetDouble
DEAL_SYMBOL	Símbolo de transacción	HistoryDealGetString
DEAL_TIME	Hora de ejecución de transacción	HistoryDealGetInteger
DEAL_TIME_MSC	Tiempo de ejecución de la transacción en milisegundos desde 01.01.1970	HistoryDealGetInteger
DEAL_TYPE	Tipo de transacción	HistoryDealGetInteger
DEAL_TYPE_BALANCE	Balance	HistoryDealGetInteger
DEAL_TYPE_BONUS	Bonos	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
DEAL_TYPE_BUY	Compra	HistoryDealGetInteger
DEAL_TYPE_BUY_CANCELED	Transacción de compra cancelada. Puede surgir la situación cuando una transacción de compra realizada anteriormente se cancele. En este caso el tipo de la operación realizada (DEAL_TYPE_BUY) se cambia a DEAL_TYPE_BUY_CANCELED, y su beneficio/pérdida se anula. El beneficio/pérdida producido/a anteriormente	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	se carga/sequita de la cuenta por medio de una operación contable separada	
DEAL_TYPE_CHARGE	Cargas adicionales	HistoryDealGetInteger
DEAL_TYPE_COMMISSION	Comisiones adicionales	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_DAILY	Comisión de agente calculada al final de la jornada de trading	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Comisión de agente calculada al final del mes	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_DAILY	Comisión calculada al final de la jornada de trading	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
DEAL_TYPE_COMMISSION_MONTHLY	Comisión calculada al final del mes	HistoryDealGetInteger
DEAL_TYPE_CORRECTION	Corrección	HistoryDealGetInteger
DEAL_TYPE_CREDIT	Crédito	HistoryDealGetInteger
DEAL_TYPE_INTEREST	Calculo de intereses sobre fondos libres	HistoryDealGetInteger
DEAL_TYPE_SELL	Venta	HistoryDealGetInteger
DEAL_TYPE_SELL_CANCELED	Transacción de venta cancelada. Puede surgir la situación cuando una transacción de venta realizada anteriormente se cancele. En este caso el tipo de la operación realizada (DEAL_TYPE_SELL) se cambia	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	a DEAL_TYPE_SELL_CANCELED, y su beneficio/pérdida se anula. El beneficio/pérdida producido/a anteriormente se carga/sequita de la cuenta por medio de una operación contable separada	
DEAL_VOLUME	Volumen de transacción	HistoryDealGetDouble
DRAW_ARROW	Dibujo de flechas	Estilos de dibujo
DRAW_BARS	Visualización como secuencia de barras	Estilos de dibujo
DRAW_CANDLES	Visualización	Estilos de dibujo

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	como secuencia de velas	
DRAW_COLOR_ARROW	Dibujo con flechas multicolor	Estilos de dibujo
DRAW_COLOR_BARS	Barras multicolor	Estilos de dibujo
DRAW_COLOR_CANDLES	Velas multicolor	Estilos de dibujo
DRAW_COLOR_HISTOGRAM	Histograma multicolor desde la línea cero	Estilos de dibujo
DRAW_COLOR_HISTOGRAM2	Histograma multicolor de dos buffers de indicadores	Estilos de dibujo
DRAW_COLOR_LINE	Línea multicolor	Estilos de dibujo
DRAW_COLOR_SECTION	Secciones multicolor	Estilos de dibujo
DRAW_COLOR_ZIGZAG	ZigZag multicolor	Estilos de dibujo
DRAW_FILLING	Relleno de color	Estilos de dibujo

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	entre dos niveles	
DRAW_HISTOGRAM	Histograma de la línea cero	Estilos de dibujo
DRAW_HISTOGRAM2	Histograma de dos buffers de indicadores	Estilos de dibujo
DRAW_LINE	Línea	Estilos de dibujo
DRAW_NONE	No se dibuja	Estilos de dibujo
DRAW_SECTION	Sección	Estilos de dibujo
DRAW_ZIGZAG	Estilo Zigzag permite secciones verticales en la barra	Estilos de dibujo
ELLIOTT_CYCLE	Ciclo (Cycle)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_GRAND_SUPERCYCLE	Gran supercicl o (Grand Supercyc le)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_INTERMEDIATE	Interme dio (Interme diate)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINOR	Menor (Minor)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUETTE	Minuette (Minuett	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	e)	
ELLIOTT_MINUTE	Minute (Minute)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_PRIMARY	Primario (Primary)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUBMINUETTE	Subminuette (Subminuette)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUPERCYCLE	Supercicl o (Supercycle)	ObjectSetInteger, ObjectGetInteger
EMPTY_VALUE	Valor vacío en el búfer de indicadores	Otras constantes
ERR_ACCOUNT_WRONG_PROPERTY	Identificador erróneo de la propiedad de la cuenta	GetLastError
ERR_ARRAY_BAD_SIZE	Tamaño solicitado del array supera 2 gigabytes	GetLastError
ERR_ARRAY_RESIZE_ERROR	No hay memoria suficiente para la reubicación de un array, o un	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	intento de cambio del tamaño de un array estático	
ERR_BOOKS_CANNOT_ADD	No se puede añadir la profundidad de mercado	GetLastError
ERR_BOOKS_CANNOT_DELETE	No se puede eliminar la profundidad de mercado	GetLastError
ERR_BOOKS_CANNOT_GET	No se puede obtener los datos de la profundidad de mercado	GetLastError
ERR_BOOKS_CANNOT_SUBSCRIBE	Error al suscribirse a la recepción de nuevos datos de la profundidad de mercado	GetLastError
ERR_BUFFERS_NO_MEMORY	No hay memoria suficiente para la redistrib	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	ución de buffers de indicadores	
ERR_BUFFERS_WRONG_INDEX	Índice erróneo de su búfer de indicadores	GetLastError
ERR_CANNOT_CLEAN_DIRECTORY	No se puede limpiar el directorio (tal vez, uno o más archivos estén bloqueados y no se ha podido llevar a cabo la eliminación)	GetLastError
ERR_CANNOT_DELETE_DIRECTORY	No se puede eliminar el directorio	GetLastError
ERR_CANNOT_DELETE_FILE	Error al eliminar el archivo	GetLastError
ERR_CANNOT_OPEN_FILE	Error al abrir el archivo	GetLastError
ERR_CHAR_ARRAY_ONLY	Tiene que ser	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	un array del tipo char	
ERR_CHART_CANNOT_CHANGE	Error al cambiar el símbolo y período del gráfico	GetLastError
ERR_CHART_CANNOT_CREATE_TIMER	Error al crear el temporizador	GetLastError
ERR_CHART_CANNOT_OPEN	Error al abrir el gráfico	GetLastError
ERR_CHART_INDICATOR_CANNOT_ADD	Error al insertar un indicador en el gráfico	GetLastError
ERR_CHART_INDICATOR_CANNOT_DEL	Error al quitar un indicador desde el gráfico	GetLastError
ERR_CHART_INDICATOR_NOT_FOUND	El indicador no ha sido encontrado en el gráfico especificado	GetLastError
ERR_CHART_NAVIGATE_FAILED	error de navegaci	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	ón por el gráfico	
ERR_CHART_NO_EXPERT	Gráfico no tiene un Asesor Experto que pueda procesar el evento	GetLastError
ERR_CHART_NO_REPLY	Gráfico no responde	GetLastError
ERR_CHART_NOT_FOUND	Gráfico no encontrado	GetLastError
ERR_CHART_SCREENSHOT_FAILED	Error al crear un screenshot	GetLastError
ERR_CHART_TEMPLATE_FAILED	Error al aplicar una plantilla	GetLastError
ERR_CHART_WINDOW_NOT_FOUND	Subventana que contiene el indicador especificado no encontrada	GetLastError
ERR_CHART_WRONG_ID	Identificador erróneo del gráfico	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ERR_CHART_WRONG_PARAMETER	Valor erróneo del parámetro para la función de trabajo con los gráficos	GetLastError
ERR_CHART_WRONG_PROPERTY	Identificador erróneo de la propiedad del gráfico	GetLastError
ERR_CUSTOM_WRONG_PROPERTY	Identificador erróneo de la propiedad del indicador personalizado	GetLastError
ERR_DIRECTORY_NOT_EXIST	Directorio no existe	GetLastError
ERR_DOUBLE_ARRAY_ONLY	Tiene que ser un array del tipo double	GetLastError
ERR_FILE_BINSTRINGSIZE	Hay que especificar el tamaño de la cadena porque el archivo	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	ha sido abierto como binario	
ERR_FILE_CACHEBUFFER_ERROR	Memoria insuficiente para la caché de lectura	GetLastError
ERR_FILE_CANNOT_REWRITE	No se puede reescribir el archivo	GetLastError
ERR_FILE_IS_DIRECTORY	No es un archivo, es un directorio	GetLastError
ERR_FILE_ISNOT_DIRECTORY	Es un archivo, no es un directorio	GetLastError
ERR_FILE_NOT_EXIST	Archivo no existe	GetLastError
ERR_FILE_NOTBIN	El archivo debe ser abierto como un archivo binario	GetLastError
ERR_FILE_NOTCSV	El archivo debe ser abierto como un archivo CSV	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ERR_FILE_NOTTOREAD	El archivo debe ser abierto para la lectura	GetLastError
ERR_FILE_NOTTOWRITE	El archivo debe ser abierto para la escritura	GetLastError
ERR_FILE_NOTTXT	El archivo debe ser abierto como un archivo de texto	GetLastError
ERR_FILE_NOTTXTORCSV	El archivo debe ser abierto como un archivo de texto o CSV	GetLastError
ERR_FILE_READERROR	Error de lectura de archivo	GetLastError
ERR_FILE_WRITEERROR	No se ha podido escribir el recurso en el archivo	GetLastError
ERR_FLOAT_ARRAY_ONLY	Tiene que ser un array del tipo float	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ERR_FTP_SEND_FAILED	Envío de archivo a través de ftp fallido	GetLastError
ERR_FUNCTION_NOT_ALLOWED	Función de sistema no está permitida para la llamada	GetLastError
ERR_GLOBALVARIABLE_EXISTS	Variable global del terminal de cliente con este nombre ya existe	GetLastError
ERR_GLOBALVARIABLE_NOT_FOUND	Variable global del terminal de cliente no encontrada	GetLastError
ERR_HISTORY_NOT_FOUND	Historial solicitado no encontrado	GetLastError
ERR_HISTORY_WRONG_PROPERTY	Identificador erróneo de la propiedad del historial	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ERR_INCOMPATIBLE_ARRAYS	Copiado de los arrays incompatibles. Un array de cadena puede ser copiado sólo en un array de cadena, un array numérico sólo en un array numérico	GetLastError
ERR_INCOMPATIBLE_FILE	Para los arrays de cadenas - un archivo de texto, para los demás - un archivo binario	GetLastError
ERR_INDICATOR_CANNOT_ADD	Error al añadir indicador	GetLastError
ERR_INDICATOR_CANNOT_APPLY	Indicador no puede ser aplicado a otro indicador	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ERR_INDICATOR_CANNOT_CREATE	No se puede crear indicador	GetLastError
ERR_INDICATOR_CUSTOM_NAME	El primer parámetro en la matriz tiene que ser el nombre del indicador personalizado	GetLastError
ERR_INDICATOR_DATA_NOT_FOUND	Datos solicitados no encontrados	GetLastError
ERR_INDICATOR_NO_MEMORY	Memoria insuficiente para añadir el indicador	GetLastError
ERR_INDICATOR_PARAMETER_TYPE	Tipo erróneo del parámetro en la matriz al crear un indicador	GetLastError
ERR_INDICATOR_PARAMETERS_MISSING	No hay parámetros cuando se crea un	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	indicador	
ERR_INDICATOR_UNKNOWN_SYMBOL	Símbolo desconocido	GetLastError
ERR_INDICATOR_WRONG_HANDLE	Handle del indicador es erróneo	GetLastError
ERR_INDICATOR_WRONG_INDEX	Índice del búfer de indicador que se solicita es erróneo	GetLastError
ERR_INDICATOR_WRONG_PARAMETERS	Número erróneo de parámetros al crear un indicador	GetLastError
ERR_INT_ARRAY_ONLY	Tiene que ser un array del tipo int	GetLastError
ERR_INTERNAL_ERROR	Error interno inesperado	GetLastError
ERR_INVALID_ARRAY	Array del tipo inapropiado, tamaño inapropiado o objeto	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	dañado del array dinámico	
ERR_INVALID_DATETIME	Valor de fecha y/o hora incorrecto	GetLastError
ERR_INVALID_FILEHANDLE	Archivo con este manejador ya está cerrado, o no se abría en absoluto	GetLastError
ERR_INVALID_PARAMETER	Parámetro erróneo durante la llamada a la función de sistema	GetLastError
ERR_INVALID_POINTER	Puntero erróneo	GetLastError
ERR_INVALID_POINTER_TYPE	Tipo erróneo del puntero	GetLastError
ERR_LONG_ARRAY_ONLY	Tiene que ser un array del tipo long	GetLastError
ERR_MAIL_SEND_FAILED	Envío de carta fallido	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ERR_MARKET_LASTTIME_UNKNOWN	Hora del último tick no se conoce (no había ticks)	GetLastError
ERR_MARKET_NOT_SELECTED	Símbolo no está seleccionado en Market Watch	GetLastError
ERR_MARKET_SELECT_ERROR	Error al agregar o eliminar el símbolo a/de Market Watch	GetLastError
ERR_MARKET_UNKNOWN_SYMBOL	Símbolo desconocido	GetLastError
ERR_MARKET_WRONG_PROPERTY	Identificador erróneo de la propiedad del símbolo	GetLastError
ERR_MQL5_WRONG_PROPERTY	Identificador erróneo de la propiedad del programa	GetLastError
ERR_NO_STRING_DATE	No hay fecha en	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	la cadena	
ERR_NOT_ENOUGH_MEMORY	No hay memoria suficiente para ejecutar la función de sistema	GetLastError
ERR_NOTIFICATION_SEND_FAILED	No se ha podido enviar la notificación	GetLastError
ERR_NOTIFICATION_TOO_FREQUENT	Envío de notificaciones muy frecuente	GetLastError
ERR_NOTIFICATION_WRONG_PARAMETER	Parámetro incorrecto para el envío de la notificación - en la función SendNotification() han pasado una línea vacía o NULL	GetLastError
ERR_NOTIFICATION_WRONG_SETTINGS	Ajustes incorrectos de las notificaciones en	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	el terminal (ID no especificada o permiso no concedido)	
ERR_NOTINITIALIZED_STRING	Cadena no inicializada	GetLastError
ERR_NUMBER_ARRAYS_ONLY	Tiene que ser un array numérico	GetLastError
ERR_OBJECT_ERROR	Error al manejar un objeto gráfico	GetLastError
ERR_OBJECT_GETDATE_FAILED	Imposible recibir fecha correspondiente al valor	GetLastError
ERR_OBJECT_GETVALUE_FAILED	Imposible recibir valor correspondiente a la fecha	GetLastError
ERR_OBJECT_NOT_FOUND	Objeto gráfico no encontrado	GetLastError
ERR_OBJECT_WRONG_PROPERTY	Identificador	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	erróneo de la propiedad del objeto gráfico	
ERR_ONEDIM_ARRAYS_ONLY	Tiene que ser un array unidimensional	GetLastError
ERR_OPENCL_BUFFER_CREATE	Error de creación del búfer OpenCL	GetLastError
ERR_OPENCL_CONTEXT_CREATE	Error al crear el contexto OpenCL	GetLastError
ERR_OPENCL_EXECUTE	Error de ejecución del programa OpenCL	GetLastError
ERR_OPENCL_INTERNAL	Error interno al ejecutar OpenCL	GetLastError
ERR_OPENCL_INVALID_HANDLE	Manejado OpenCL incorrecto	GetLastError
ERR_OPENCL_KERNEL_CREATE	Error al crear el kernel - punto de entrada de OpenCL	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ERR_OPENCL_NOT_SUPPORTED	Las funciones OpenCL no se soportan en este ordenador	GetLastError
ERR_OPENCL_PROGRAM_CREATE	Error al compilar el programa OpenCL	GetLastError
ERR_OPENCL_QUEUE_CREATE	Error al crear la cola de ejecución en OpenCL	GetLastError
ERR_OPENCL_SET_KERNEL_PARAMETER	Error al establecer los parámetros para el kernel OpenCL (punto de entrada en el programa OpenCL)	GetLastError
ERR_OPENCL_TOO_LONG_KERNEL_NAME	Punto de entrada demasiado largo (kernel OpenCL)	GetLastError
ERR_OPENCL_WRONG_BUFFER_OFFSET	Desplazamiento incorrecto en el	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	búfer OpenCL	
ERR_OPENCL_WRONG_BUFFER_SIZE	Tamaño del búfer OpenCL incorrecto	GetLastError
ERR_PLAY_SOUND_FAILED	Reproducción de sonido fallido	GetLastError
ERR_RESOURCE_NAME_DUPLICATED	Coincidencia del nombre del recurso dinámico y estático	GetLastError
ERR_RESOURCE_NAME_IS_TOO_LONG	Nombre del recurso supera 63 caracteres	GetLastError
ERR_RESOURCE_NOT_FOUND	Recurso con este nombre no encontrado en EX5	GetLastError
ERR_RESOURCE_UNSUPPORTED_TYPE	Tipo del recurso no soportado o el tamaño superior a 16 Mb	GetLastError
ERR_SERIES_ARRAY	No se puede	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	usar serie temporal	
ERR_SHORT_ARRAY_ONLY	Tiene que ser un array del tipo short	GetLastError
ERR_SMALL_ARRAY	Un array muy pequeño, posición de inicio está fuera de los límites del array	GetLastError
ERR_SMALL_ASSERIES_ARRAY	El array que recibe está declarado como AS_SERIES, y no tiene el tamaño suficiente	GetLastError
ERR_STRING_OUT_OF_MEMORY	Memoria insuficiente para la cadena	GetLastError
ERR_STRING_RESIZE_ERROR	No hay memoria suficiente para la reubicación de una cadena	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ERR_STRING_SMALL_LEN	Longitud de cadena es menos de la esperada	GetLastError
ERR_STRING_TIME_ERROR	Error de conversión de cadena a fecha	GetLastError
ERR_STRING_TOO_BIGNUMBER	Número excesivamente grande, más que ULONG_MAX	GetLastError
ERR_STRING_UNKNOWNTYPE	Tipo de datos desconocido durante la conversión a una cadena	GetLastError
ERR_STRING_ZEROADDED	Al final de la cadena se ha añadido 0, una operación inútil	GetLastError
ERR_STRINGPOS_OUTOFRANGE	Posición fuera de los límites de la cadena	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ERR_STRUCT_WITHOBJECTS_ORCLASS	Estructura contiene objetos de cadenas y/o de arrays dinámicos y/o estructuras con estos objetos y/o clases	GetLastError
ERR_SUCCESS	Operación ejecutada con éxito	GetLastError
ERR_TERMINAL_WRONG_PROPERTY	Identificador erróneo de la propiedad del terminal	GetLastError
ERR_TOO_LONG_FILENAME	Nombre del archivo demasiado largo	GetLastError
ERR_TOO_MANY_FILES	No se puede abrir más de 64 archivos	GetLastError
ERR_TOO_MANY_FORMATTERS	Hay más especificadores de formato	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	que los parámetros	
ERR_TOO_MANY_PARAMETERS	Hay más Parámetros que los especificadores de formato	GetLastError
ERR_TRADE_DEAL_NOT_FOUND	Transacción no encontrada	GetLastError
ERR_TRADE_DISABLED	Prohibida la actividad comercial para el Asesor Experto	GetLastError
ERR_TRADE_ORDER_NOT_FOUND	Orden no encontrada	GetLastError
ERR_TRADE_POSITION_NOT_FOUND	Posición no encontrada	GetLastError
ERR_TRADE_SEND_FAILED	Envío de solicitud comercial fallida	GetLastError
ERR_TRADE_WRONG_PROPERTY	Identificador erróneo de la propiedad de la	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	actividad comercial	
ERR_USER_ERROR_FIRST	A partir de este código se empiezan los errores definidos por el usuario	GetLastError
ERR_WEBREQUEST_CONNECT_FAILED	No se ha podido conectar se a la URL especificada	GetLastError
ERR_WEBREQUEST_INVALID_ADDRESS	URL no ha superado la prueba	GetLastError
ERR_WEBREQUEST_REQUEST_FAILED	Error de ejecución de la solicitud HTTP	GetLastError
ERR_WEBREQUEST_TIMEOUT	Superado el tiempo de espera de recepción de datos	GetLastError
ERR_WRONG_DIRECTORYNAME	Nombre erróneo del	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	directorio	
ERR_WRONG_FILEHANDLE	Manejador erróneo de archivo	GetLastError
ERR_WRONG_FILENAME	Nombre del archivo no válido	GetLastError
ERR_WRONG_FORMATSTRING	Cadena de formato errónea	GetLastError
ERR_WRONG_INTERNAL_PARAMETER	Parámetro erróneo durante la llamada built-in a la función del terminal de cliente	GetLastError
ERR_WRONG_STRING_DATE	Fecha errónea en la cadena	GetLastError
ERR_WRONG_STRING_OBJECT	Objeto de cadena dañado	GetLastError
ERR_WRONG_STRING_PARAMETER	Parámetro del tipo string dañado	GetLastError

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ERR_WRONG_STRING_TIME	Hora errónea en la cadena	GetLastError
ERR_ZEROSIZE_ARRAY	Un array de longitud cero	GetLastError
FILE_ACCESS_DATE	Fecha del último acceso al archivo	FileGetInteger
FILE_ANSI	Cadenas del tipo ANSI (símbolos de un byte). La bandera se usa cuando un archivo se abre, en la función (FileOpen())	FileOpen
FILE_BIN	Modo binario de lectura-escritura (sin conversión de una cadena y en una cadena). La bandera	FileOpen

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	se usa cuando un archivo se abre, en la función (FileOpen())	
FILE_COMMON	Ubicación del archivo en la carpeta general de todos los terminales de cliente \Terminal\Common\Files. La bandera se usa en las funciones (FileOpen()), (FileCopy(), FileMove()), (FileIsExist())	FileOpen, FileCopy, FileMove, FileIsExist
FILE_CREATE_DATE	Fecha de creación	FileGetInteger
FILE_CSV	Archivo del tipo csv (todos sus elementos se	FileOpen

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	convierten a las cadenas del tipo correspondiente, unicode o ansi, y se separan con un delimitador). La bandera se usa cuando un archivo se abre, en la función (FileOpen())	
FILE_END	Obtención del signo del final del archivo	FileGetInteger
FILE_EXISTS	Comprobación de existencia	FileGetInteger
FILE_IS_ANSI	El archivo está abierto como un archivo ANSI (ver FILE_ANSI)	FileGetInteger
FILE_IS_BINARY	El archivo está	FileGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	abierto como un archivo binario (ver FILE_BIN)	
FILE_IS_COMMON	El archivo está abierto en la carpeta común de todos los terminales de cliente (ver FILE_COMMON)	FileGetInteger
FILE_IS_CSV	El archivo está abierto como un archivo CSV (ver FILE_CSV)	FileGetInteger
FILE_IS_READABLE	El archivo está abierto con posibilidades de lectura (ver FILE_READ)	FileGetInteger
FILE_IS_TEXT	El archivo está	FileGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	abierto como un archivo de texto (ver FILE_TXT)	
FILE_IS_WRITABLE	El archivo está abierto con posibilidades de escritura (ver FILE_WRITE)	FileGetInteger
FILE_LINE_END	Obtención del signo del final de la línea	FileGetInteger
FILE_MODIFY_DATE	Fecha de última modificación	FileGetInteger
FILE_POSITION	Posición del puntero en el archivo	FileGetInteger
FILE_READ	Archivo se abre para la lectura. La bandera se usa cuando un archivo se abre, en la función	FileOpen

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	(FileOpen()). Cuando abrimos un archivo, es obligatorio indicar la bandera FILE_WRITE y/o la bandera FILE_READ.	
FILE_REWRITE	Posibilidad de reescribir un archivo usando las funciones FileCopy() y FileMove(). El archivo tiene que existir o abrirse para la escritura. En caso contrario el archivo no se abrirá.	FileCopy, FileMove

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
FILE_SHARE_READ	Acceso compartido para la lectura desde varios programas. La bandera se usa cuando se abre un archivo (FileOpen()), pero durante la apertura no sustituye la necesidad de indicar FILE_WRITE y/o la bandera FILE_READ	FileOpen
FILE_SHARE_WRITE	Acceso compartido para la escritura desde varios programas. La bandera se usa cuando se abre un	FileOpen

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	archivo (FileOpen()), pero durante la apertura no sustituye la necesidad de indicar FILE_WRITE y/o la bandera FILE_READ	
FILE_SIZE	Tamaño del archivo en bytes	FileGetInteger
FILE_TXT	Archivo de texto simple (igual que el archivo csv pero sin tomar en cuenta los delimitadores). La bandera se usa cuando un archivo se abre, en la función	FileOpen

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	(FileOpen())	
FILE_UNICODE	Cadenas del tipo UNICODE (símbolos de dos byte). La bandera se usa cuando un archivo se abre, en la función (FileOpen())	FileOpen
FILE_WRITE	Archivo se abre para la escritura. La bandera se usa cuando un archivo se abre, en la función (FileOpen()). Cuando abrimos un archivo, es obligatorio indicar la bandera FILE_WRITE y/o	FileOpen

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	la bandera FILE_READ.	
FLT_DIG	Número de dígitos decimales significativos	Constantes de tipos numéricos
FLT_EPSILON	Valor mínimo que satisface la condición: 1.0+FLT_EPSILON != 1.0	Constantes de tipos numéricos
FLT_MANT_DIG	Cantidad de bits en la mantisa	Constantes de tipos numéricos
FLT_MAX	Valor máximo que puede ser representado por el tipo float	Constantes de tipos numéricos
FLT_MAX_10_EXP	Valor decimal máximo del grado de exponente	Constantes de tipos numéricos
FLT_MAX_EXP	Valor binario máximo	Constantes de tipos numéricos

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	del grado de exponente	
FLT_MIN	Valor mínimo positivo que puede ser representado por el tipo float	Constantes de tipos numéricos
FLT_MIN_10_EXP	Valor decimal mínimo del grado de exponente	Constantes de tipos numéricos
FLT_MIN_EXP	Valor binario mínimo del grado de exponente	Constantes de tipos numéricos
FRIDAY	Viernes	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
GANN_DOWN_TREND	Línea de tendencia a la baja	ObjectSetInteger, ObjectGetInteger
GANN_UP_TREND	Línea de tendencia al alza	ObjectSetInteger, ObjectGetInteger
GATORJAW_LINE	Línea de mandíbulas	Líneas de indicadores
GATORLIPS_LINE	Línea de labios	Líneas de indicadores

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
GATORTEETH_LINE	Línea de dientes	Líneas de indicadores
IDABORT	El botón "Interrumpir" (Abort) está seleccionado	MessageBox
IDCANCEL	El botón "Cancelar" (Cancel) está seleccionado	MessageBox
IDCONTINUE	El botón "Continuar" (Continue) está seleccionado	MessageBox
IDIGNORE	El botón "Ignorar" (Ignore) está seleccionado	MessageBox
IDNO	El botón "No" (No) está seleccionado	MessageBox
IDOK	El botón "OK" está seleccionado	MessageBox
IDRETRY	El botón "Reintentar" (Retry) está	MessageBox

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	seleccionado	
IDTRYAGAIN	El botón "Repetir" (Try Again) está seleccionado	MessageBox
IDYES	El botón "Sí" (Yes) está seleccionado	MessageBox
IND_AC	Accelerator Oscillator	IndicatorCreate, IndicatorParameters
IND_AD	Accumulation/Distribution	IndicatorCreate, IndicatorParameters
IND_ADX	Average Directional Index	IndicatorCreate, IndicatorParameters
IND_ADXW	ADX by Welles Wilder	IndicatorCreate, IndicatorParameters
IND_ALLIGATOR	Alligator	IndicatorCreate, IndicatorParameters
IND_AMA	Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_AO	Awesome Oscillator	IndicatorCreate, IndicatorParameters
IND_ATR	Average True Range	IndicatorCreate, IndicatorParameters

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
IND_BANDS	Bollinger Bands®	IndicatorCreate, IndicatorParameters
IND_BEARS	Bears Power	IndicatorCreate, IndicatorParameters
IND_BULLS	Bulls Power	IndicatorCreate, IndicatorParameters
IND_BWMFI	Market Facilitation Index	IndicatorCreate, IndicatorParameters
IND_CCI	Commodity Channel Index	IndicatorCreate, IndicatorParameters
IND_CHAIKIN	Chaikin Oscillator	IndicatorCreate, IndicatorParameters
IND_CUSTOM	Custom indicator	IndicatorCreate, IndicatorParameters
IND_DEMA	Double Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_DEMARKER	DeMarker	IndicatorCreate, IndicatorParameters
IND_ENVELOPES	Envelopes	IndicatorCreate, IndicatorParameters
IND_FORCE	Force Index	IndicatorCreate, IndicatorParameters
IND_FRACTALS	Fractals	IndicatorCreate, IndicatorParameters
IND_FRAMA	Fractal Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_GATOR	Gator Oscillator	IndicatorCreate, IndicatorParameters
IND_ICHIMOKU	Ichimoku Kinko	IndicatorCreate, IndicatorParameters

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	Hyo	
IND_MA	Moving Average	IndicatorCreate, IndicatorParameters
IND_MACD	MACD	IndicatorCreate, IndicatorParameters
IND_MFI	Money Flow Index	IndicatorCreate, IndicatorParameters
IND_MOMENTUM	Momentum	IndicatorCreate, IndicatorParameters
IND_OBV	On Balance Volume	IndicatorCreate, IndicatorParameters
IND_OSMA	OsMA	IndicatorCreate, IndicatorParameters
IND_RSI	Relative Strength Index	IndicatorCreate, IndicatorParameters
IND_RVI	Relative Vigor Index	IndicatorCreate, IndicatorParameters
IND_SAR	Parabolic SAR	IndicatorCreate, IndicatorParameters
IND_STDDEV	Standard Deviation	IndicatorCreate, IndicatorParameters
IND_STOCHASTIC	Stochastic Oscillator	IndicatorCreate, IndicatorParameters
IND_TEMA	Triple Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_TRIX	Triple Exponential Moving Averages	IndicatorCreate, IndicatorParameters

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	Oscillator	
IND_VIDYA	Variable Index Dynamic Average	IndicatorCreate, IndicatorParameters
IND_VOLUMES	Volumes	IndicatorCreate, IndicatorParameters
IND_WPR	Williams' Percent Range	IndicatorCreate, IndicatorParameters
INDICATOR_CALCULATIONS	Buffers auxiliares para los cálculos intermedios	SetIndexBuffer
INDICATOR_COLOR_INDEX	Colores	SetIndexBuffer
INDICATOR_DATA	Datos para dibujar	SetIndexBuffer
INDICATOR_DIGITS	Precisión de dibujo de los valores del indicador	IndicatorSetInteger
INDICATOR_HEIGHT	El alto fijo de la propia ventana del indicador (comando del preprocesador #property	IndicatorSetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	indicator_height)	
INDICATOR_LEVELCOLOR	Color de la línea del nivel	IndicatorSetInteger
INDICATOR_LEVELS	Número de niveles en la ventana del indicador	IndicatorSetInteger
INDICATOR_LEVELSTYLE	Estilo de la línea del nivel	IndicatorSetInteger
INDICATOR_LEVELTEXT	Descripción del nivel	IndicatorSetString
INDICATOR_LEVELVALUE	Valor del nivel	IndicatorSetDouble
INDICATOR_LEVELWIDTH	Grosor de la línea del nivel	IndicatorSetInteger
INDICATOR_MAXIMUM	Máximo de la ventana del indicador	IndicatorSetDouble
INDICATOR_MINIMUM	Mínimo de la ventana del indicador	IndicatorSetDouble
INDICATOR_SHORTNAME	Nombre breve del indicador	IndicatorSetString

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
INT_MAX	Valor máximo que puede ser representado por el tipo int	Constantes de tipos numéricos
INT_MIN	Valor mínimo que puede ser representado por el tipo int	Constantes de tipos numéricos
INVALID_HANDLE	Manejador incorrecto	Otras constantes
IS_DEBUG_MODE	Indica que un programa mql5 se encuentra en el modo de depuración	Otras constantes
IS_PROFILE_MODE	Indica que un programa mql5 se encuentra en el modo de perfilación	Otras constantes
KIJUNSEN_LINE	Línea Kijun-sen	Líneas de indicadores

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
LICENSE_DEMO	Versión demo del producto de pago de la Tienda. Funciona sólo en el Probador de Estrategias	MQLInfoInteger
LICENSE_FREE	Versión gratuita ilimitada	MQLInfoInteger
LICENSE_FULL	Esta es una versión con licencia adquirida, que permite al menos 5 activaciones. El vendedor podrá aumentar el número permitido de activaciones.	MQLInfoInteger
LICENSE_TIME	Esta es una versión con un término de licencia limitado	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
LONG_MAX	Valor máximo que puede ser representado por el tipo long	Constantes de tipos numéricos
LONG_MIN	Valor mínimo que puede ser representado por el tipo long	Constantes de tipos numéricos
LOWER_BAND	Límite inferior	Líneas de indicadores
LOWER_HISTOGRAM	Histograma de abajo	Líneas de indicadores
LOWER_LINE	Línea inferior	Líneas de indicadores
M_1_PI	1/pi	Constantes matemáticas
M_2_PI	2/pi	Constantes matemáticas
M_2_SQRTPI	2/sqrt(pi)	Constantes matemáticas
M_E	e	Constantes matemáticas
M_LN10	ln(10)	Constantes matemáticas
M_LN2	ln(2)	Constantes matemáticas
M_LOG10E	log10(e)	Constantes matemáticas
M_LOG2E	log2(e)	Constantes matemáticas
M_PI	pi	Constantes matemáticas
M_PI_2	pi/2	Constantes matemáticas
M_PI_4	pi/4	Constantes matemáticas

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
M_SQRT1_2	1/sqrt(2)	Constantes matemáticas
M_SQRT2	sqrt(2)	Constantes matemáticas
MAIN_LINE	Línea principal	Líneas de indicadores
MB_ABORTRETRYIGNORE	La ventana de diálogo contiene tres botones: Abort, Retry y Ignore	MessageBox
MB_CANCELTRYCONTINUE	La ventana de diálogo contiene tres botones: Cancel, Try Again, Continue	MessageBox
MB_DEFBUTTON1	El primer botón MB_DEFBUTTON1 - el botón está seleccionado por defecto, si MB_DEFBUTTON2, MB_DEFBUTTON3, o MB_DEF	MessageBox

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	BUTTON4 no están especificados	
MB_DEFBUTTON2	El segundo botón - botón por defecto	MessageBox
MB_DEFBUTTON3	El tercer botón - botón por defecto	MessageBox
MB_DEFBUTTON4	El cuarto botón - botón por defecto	MessageBox
MB_ICONEXCLAMATION, MB_ICONWARNING	El ícono del signo de admiración	MessageBox
MB_ICONINFORMATION, MB_ICONASTERISK	El ícono del signo i dentro del círculo	MessageBox
MB_ICONQUESTION	El ícono del signo interrogación	MessageBox
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	El ícono del signo STOP	MessageBox
MB_OK	La ventana de diálogo contiene	MessageBox

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	un botón: OK. Por defecto	
MB_OKCANCEL	La ventana de diálogo contiene dos botones: OK y Cancel	MessageBox
MB_RETRYCANCEL	La ventana de diálogo contiene dos botones: Retry y Cancel	MessageBox
MB_YESNO	La ventana de diálogo contiene dos botones: Yes y No	MessageBox
MB_YESNOCANCEL	La ventana de diálogo contiene tres botones: Yes, No y Cancel	MessageBox
MINUSDI_LINE	Línea -DI	Líneas de indicadores
MODE_EMA	Media móvil	Métodos de alisamiento

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	exponencial	
MODE_LWMA	Media móvil ponderada	Métodos de alisamiento
MODE_SMA	Media móvil simple	Métodos de alisamiento
MODE_SMMA	Media móvil suavizada	Métodos de alisamiento
MONDAY	Lunes	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
MQL_DEBUG	Indica que el programa iniciado trabaja en el modo de depuración	MQLInfoInteger
MQL_DLLS_ALLOWED	Permiso de usar DLL para este programa iniciado	MQLInfoInteger
MQL_FRAME_MODE	Indica que el EA iniciado en el gráfico trabaja en el modo de recogida	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	de los frames de resultados de la optimización	
MQL_LICENSE_TYPE	Tipo de licencia del módulo EX5. La licencia se refiere al módulo EX5 desde el cual se hace la solicitud con el uso de MQLInfoInteger(MQL_LICENSE_TYPE).	MQLInfoInteger
MQL_MEMORY_LIMIT	Tamaño máximo posible de la memoria dinámica para el programa MQL5 en Mb	MQLInfoInteger
MQL_MEMORY_USED	Tamaño de la memoria usada por el program	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	a MQL5 en Mb	
MQL_OPTIMIZATION	Indica que el programa iniciado trabaja en el proceso de optimización	MQLInfoInteger
MQL_PROFILER	Indica que el programa iniciado trabaja en el modo de perfiladura del código	MQLInfoInteger
MQL_PROGRAM_NAME	Nombre del programa mql5 en ejecución	MQLInfoString
MQL_PROGRAM_PATH	Ruta para dicho programa en ejecución	MQLInfoString
MQL_PROGRAM_TYPE	Tipo del programa mql5	MQLInfoInteger
MQL_SIGNALS_ALLOWED	Permiso para trabajar	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	con las señales de este programa iniciado	
MQL_TESTER	Indica que el programa iniciado trabaja en el Probador	MQLInfoInteger
MQL_TRADE_ALLOWED	Permiso para tradear concedido a este programa iniciado	MQLInfoInteger
MQL_VISUAL_MODE	Indica que el programa iniciado trabaja en el modo visual de simulación	MQLInfoInteger
NULL	Cero de cualquier tipo	Otras constantes
OBJ_ALL_PERIODS	El objeto se dibuja en todos los períodos de tiempo	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
OBJ_ARROW	Objeto "Flecha"	Tipos de objetos
OBJ_ARROW_BUY	Signo "Buy"	Tipos de objetos
OBJ_ARROW_CHECK	Signo "Marca" (tic)	Tipos de objetos
OBJ_ARROW_DOWN	Signo "Flecha abajo"	Tipos de objetos
OBJ_ARROW_LEFT_PRICE	Etiqueta izquierda de precio	Tipos de objetos
OBJ_ARROW_RIGHT_PRICE	Etiqueta derecha de precio	Tipos de objetos
OBJ_ARROW_SELL	Signo "Sell"	Tipos de objetos
OBJ_ARROW_STOP	Signo "Stop"	Tipos de objetos
OBJ_ARROW_THUMB_DOWN	Signo "Mal" (dedo pulgar abajo)	Tipos de objetos
OBJ_ARROW_THUMB_UP	Signo "Bien" (dedo pulgar arriba)	Tipos de objetos
OBJ_ARROW_UP	Signo "Flecha arriba"	Tipos de objetos
OBJ_ARROWED_LINE	Objeto "Línea de separación con	Tipos de objetos

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	una flecha"	
OBJ_BITMAP	Objeto "Dibujo"	Tipos de objetos
OBJ_BITMAP_LABEL	Objeto "Etiqueta gráfica"	Tipos de objetos
OBJ_BUTTON	Objeto "Botón"	Tipos de objetos
OBJ_CHANNEL	Canal equidistante	Tipos de objetos
OBJ_CHART	Objeto "Gráfico"	Tipos de objetos
OBJ_CYCLES	Líneas cíclicas	Tipos de objetos
OBJ_EDIT	Objeto "Campo de edición"	Tipos de objetos
OBJ_ELLIOTWAVE3	Ondas de Elliott de fase correctiva (3)	Tipos de objetos
OBJ_ELLIOTWAVE5	Ondas de Elliott de fase impulsiva (5)	Tipos de objetos
OBJ_ELLIPSE	Elipse	Tipos de objetos
OBJ_EVENT	Objeto "Evento" corresponde a un evento en el calendario	Tipos de objetos

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	económico	
OBJ_EXPANSION	Expansión de Fibonacci	Tipos de objetos
OBJ_FIBO	Retrocesos de Fibonacci	Tipos de objetos
OBJ_FIBOARC	Arcos de Fibonacci	Tipos de objetos
OBJ_FIBOCHANNEL	Canal de Fibonacci	Tipos de objetos
OBJ_FIBOFAN	Abanico de Fibonacci	Tipos de objetos
OBJ_FIBOTIMES	Zonas temporales de Fibonacci	Tipos de objetos
OBJ_GANNFAN	Abanico de Gann	Tipos de objetos
OBJ_GANNGRID	Cuadrícula de Gann	Tipos de objetos
OBJ_GANNLINE	Líneas de Gann	Tipos de objetos
OBJ_HLINE	Línea horizontal	Tipos de objetos
OBJ_LABEL	Objeto "Etiqueta de texto"	Tipos de objetos

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
OBJ_NO_PERIODS	El objeto no se muestra en ninguno de los períodos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_D1	El objeto se dibuja en los gráficos de un día	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H1	El objeto se dibuja en los gráficos de 1 hora	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H12	El objeto se dibuja en los gráficos de 12 horas	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H2	El objeto se dibuja en los gráficos de 2 horas	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H3	El objeto se dibuja en los gráficos de 3 horas	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H4	El objeto se	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	dibuja en los gráficos de 4 horas	
OBJ_PERIOD_H6	El objeto se dibuja en los gráficos de 6 horas	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H8	El objeto se dibuja en los gráficos de 8 horas	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M1	El objeto se dibuja en los gráficos de 1 minuto	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M10	El objeto se dibuja en los gráficos de 10 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M12	El objeto se dibuja en los gráficos de 12 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M15	El objeto se dibuja en los	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	gráficos de 15 minutos	
OBJ_PERIOD_M2	El objeto se dibuja en los gráficos de 2 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M20	El objeto se dibuja en los gráficos de 20 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M3	El objeto se dibuja en los gráficos de 3 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M30	El objeto se dibuja en los gráficos de 30 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M4	El objeto se dibuja en los gráficos de 4 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M5	El objeto se dibuja en los gráficos	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	de 5 minutos	
OBJ_PERIOD_M6	El objeto se dibuja en los gráficos de 6 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_MN1	El objeto se dibuja en los gráficos mensuales	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_W1	El objeto se dibuja en los gráficos semanales	ObjectSetInteger, ObjectGetInteger
OBJ_PITCHFORK	Tridente de Andrews	Tipos de objetos
OBJ_RECTANGLE	Rectángulo	Tipos de objetos
OBJ_RECTANGLE_LABEL	Objeto "Etiqueta rectangular" sirve para crear y formatear la interfaz gráfica personalizada.	Tipos de objetos

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
OBJ_REGRESSION	Canal en regresión lineal	Tipos de objetos
OBJ_STDDEVCHANNEL	Canal de desviación estándar	Tipos de objetos
OBJ_TEXT	Objeto "Texto"	Tipos de objetos
OBJ_TREND	Línea de tendencia	Tipos de objetos
OBJ_TRENDBYANGLE	Línea de tendencia por ángulo	Tipos de objetos
OBJ_TRIANGLE	Triángulo	Tipos de objetos
OBJ_VLINE	Línea vertical	Tipos de objetos
OBJPROP_ALIGN	Alineación del texto horizontalmente en el objeto "Campo de edición" (OBJ_EDIT)	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANCHOR	Posición del punto de anclaje de un objeto gráfico	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANGLE	Ángulo. Para los	ObjectSetDouble, ObjectGetDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	objetos con el ángulo no especificado, creados desde el programa, el valor es igual a EMPTY_VALUE	
OBJPROP_ARROWCODE	Código de flecha para el objeto "Flecha"	ObjectSetInteger, ObjectGetInteger
OBJPROP_BACK	Objeto en el fondo	ObjectSetInteger, ObjectGetInteger
OBJPROP_BGCOLOR	Color del fondo para el OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	ObjectSetInteger, ObjectGetInteger
OBJPROP_BMPFILE	Nombre del archivo BMP para el objeto "Etiqueta gráfica". Véase también Recursos	ObjectSetString, ObjectGetString

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
OBJPROP_BORDER_COLOR	Color del borde para el objeto OBJ_EDIT y OBJ_BUTTON	ObjectSetInteger, ObjectGetInteger
OBJPROP_BORDER_TYPE	Estilo del borde del objeto "Etiqueta rectangular"	ObjectSetInteger, ObjectGetInteger
OBJPROP_CHART_ID	Identificador del objeto "Gráfico" (OBJ_CHART). Permite trabajar con las propiedades de este objeto como con cualquier otro gráfico usando las funciones descritas en el apartado Operaciones con gráficos, pero hay	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	algunas excepciones.	
OBJPROP_CHART_SCALE	Escala para el objeto "Gráfico"	ObjectSetInteger, ObjectGetInteger
OBJPROP_COLOR	Color	ObjectSetInteger, ObjectGetInteger
OBJPROP_CORNER	Esquina del gráfico para enlazar un objeto gráfico	ObjectSetInteger, ObjectGetInteger
OBJPROP_CREATETIME	Tiempo de creación de objeto	ObjectSetInteger, ObjectGetInteger
OBJPROP_DATE_SCALE	Visualiza la escala de tiempo para el objeto "Gráfico"	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEGREE	Nivel de marcación ondulada de Elliott	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEVIATION	Desviación para el canal de la desviación estándar	ObjectSetDouble, ObjectGetDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
OBJPROP_DIRECTION	Tendencia del objeto de Gann	ObjectSetInteger, ObjectGetInteger
OBJPROP_DRAWLINES	Visualización de líneas para la marcación ondulada de Elliott	ObjectSetInteger, ObjectGetInteger
OBJPROP_ELLIPSE	Visualización del elipse entero para el objeto "Arcos de Fibonacci" (OBJ_FIBOARC)	ObjectSetInteger, ObjectGetInteger
OBJPROP_FILL	Relleno de objeto con color (para OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	ObjectSetInteger, ObjectGetInteger
OBJPROP_FONT	Fuente	ObjectSetString, ObjectGetString

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
OBJPROP_FONTSIZE	Tamaño de caracteres	ObjectSetInteger, ObjectGetInteger
OBJPROP_HIDDEN	Prohíbe mostrar el nombre del objeto gráfico en la lista de objetos del menú del terminal "Gráficos" - "Objetos" - "Lista de objetos". El valor true permite ocultar el objeto que el usuario no necesita. Por defecto, true se pone para los objetos que muestran los eventos del calendario, historial	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	de trading, así como para los objetos creados en un programa MQL5. Para poder ver estos objetos gráfico y acceder a sus propiedades, hay que pulsar en el botón "Todos" en la ventana "Lista de objetos".	
OBJPROP_LEVELCOLOR	Color de línea-nivel	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELS	Número de niveles	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELSTYLE	Estilo de línea-nivel	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELTEXT	Descripción de nivel	ObjectSetString, ObjectGetString
OBJPROP_LEVELVALUE	Valor de nivel	ObjectSetDouble, ObjectGetDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
OBJPROP_LEVELWIDTH	Grosor de línea-nivel	ObjectSetInteger, ObjectGetInteger
OBJPROP_NAME	Nombre de objeto	ObjectSetString, ObjectGetString
OBJPROP_PERIOD	Período de tiempo para el objeto "Gráfico"	ObjectSetInteger, ObjectGetInteger
OBJPROP_PRICE	Coordenadas de precio	ObjectSetDouble, ObjectGetDouble
OBJPROP_PRICE_SCALE	Visualiza la escala de precios para el objeto "Gráfico"	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY	Línea vertical va a través de todas las ventanas del gráfico	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_LEFT	Rayo va a la izquierda	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_RIGHT	Rayo va a la derecha	ObjectSetInteger, ObjectGetInteger
OBJPROP_READONLY	Posibilidad de editar el	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	texto en el objeto Edit	
OBJPROP_SCALE	Escala (propiedad de objetos de Gann y del objeto "Arcos de Fibonacci")	ObjectSetDouble, ObjectGetDouble
OBJPROP_SELECTABLE	Disponibilidad de objeto	ObjectSetInteger, ObjectGetInteger
OBJPROP_SELECTED	Selección de objeto	ObjectSetInteger, ObjectGetInteger
OBJPROP_STATE	Estado del botón (Pulsado/Despulsado)	ObjectSetInteger, ObjectGetInteger
OBJPROP_STYLE	Estilo	ObjectSetInteger, ObjectGetInteger
OBJPROP_SYMBOL	Símbolo para el objeto "Gráfico"	ObjectSetString, ObjectGetString
OBJPROP_TEXT	Descripción de objeto (texto contenido en el objeto)	ObjectSetString, ObjectGetString
OBJPROP_TIME	Coordenadas de tiempo	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
OBJPROP_TIMEFRAMES	Visibilidad de objeto en períodos de tiempo	ObjectSetInteger, ObjectGetInteger
OBJPROP_TOOLTIP	El texto de la ayuda emergente. Si la propiedad no está establecida, entonces sale la ayuda generada automáticamente por el terminal. Puede desactivar la visualización de ayuda emergente poniendo el valor "\n" (salto de línea) para esta propiedad	ObjectSetString, ObjectGetString
OBJPROP_TYPE	Tipo de objeto	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
OBJPROP_WIDTH	Grosor de línea	ObjectSetInteger, ObjectGetInteger
OBJPROP_XDISTANCE	Distancia en píxeles por el eje X desde la esquina de enlace (ver nota)	ObjectSetInteger, ObjectGetInteger
OBJPROP_XOFFSET	Coordenada X de la esquina superior izquierda del área rectangular de visibilidad en los objetos gráficos "Etiqueta gráfica" y "Dibujo" (OBJ_BITMAP_LABEL y OBJ_BITMAP). El valor se establece en píxeles respecto a la esquina superior izquierd	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	a de la imagen original.	
OBJPROP_XSIZE	Ancho del objeto por el eje X en píxeles. Se establece para los objetos OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	ObjectSetInteger, ObjectGetInteger
OBJPROP_YDISTANCE	Distancia en píxeles por el eje Y desde la esquina de enlace (ver nota)	ObjectSetInteger, ObjectGetInteger
OBJPROP_YOFFSET	Coordenada Y de la esquina	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	superior izquierda del área rectangular de visibilidad en los objetos gráficos "Etiqueta gráfica" y "Dibujo" (OBJ_BITMAP_LABEL y OBJ_BITMAP). El valor se establece en píxeles respecto a la esquina superior izquierda de la imagen original.	
OBJPROP_YSIZE	Alto del objeto por el eje Y en píxeles. Se establece para los objetos OBJ_LABEL (read only), OBJ_BUTTON,	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_ZORDER	La prioridad de un objeto gráfico para obtener el evento de cliquear sobre el gráfico (CHART_EVENT_CLICK). Por defecto, cuando se crea un objeto, este valor se pone a cero; pero si hace falta, se puede subir la prioridad. Cuando los	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	objetos se aplican uno al otro, sólo uno de ellos, cuya prioridad es superior, recibirá el evento CHARTEVENT_CLICK.	
ORDER_COMMENT	Comentario	OrderGetString, HistoryOrderGetString
ORDER_FILLING_FOK	Esta política de ejecución significa que la orden puede ser ejecutada exclusivamente en el volumen especificado. Si en este momento en el mercado no hay volumen necesario del	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	instrumento financiero requerido, esta orden no se ejecutará. El volumen necesario puede ser cubierto por varias ofertas disponibles en este momento o en el mercado.	
ORDER_FILLING_IOC	Significa que el trader acepta realizar la transacción en el volumen máximo disponible en el mercado dentro del margen especificado en la orden. Si la ejecución	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	completa no es posible, la orden será ejecutada en el volumen disponible, y el resto del volumen no cubierto será cancelado.	
ORDER_FILLING_RETURN	Esta política de ejecución se utiliza para las órdenes de mercado (ORDER_TYPE_BUY y ORDER_TYPE_SELL), limitadas y limitadas con Stop (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BU	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	Y_STOP_LIMIT y ORDER_TYPE_SELL_STOP_LIMIT), y sólo en los modos "Ejecución por Mercado" y "Ejecución por Bolsa". En caso de la ejecución parcial, una orden limitada o de mercado con el volumen no cubierto no se anula, sino sigue estando vigente. Para las órdenes ORDER_TYPE_BUY_STOP_LIMIT y ORDER_TYPE_SELL_STOP_LIMIT en el moment	

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	o de la activación será creada la orden correspondiente limitada ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT con el tipo de ejecución ORDER_FILLING_RETURN.	
ORDER_MAGIC	Identificador del Asesor Experto que ha colocado la orden (sirve para que cada Asesor Experto ponga su único número personal)	OrderGetInteger, HistoryOrderGetInteger
ORDER_POSITION_ID	Identificador de posición que se coloca en la orden a	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	la hora de su ejecución. Cada orden ejecutada provoca una transacción que abre una nueva posición, o cambia una que ya existe. El identificador de esta misma posición se establece para la orden ejecutada en este momento.	
ORDER_PRICE_CURRENT	Precio actual del símbolo de la orden	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_OPEN	Precio especificado en la orden	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_STOPLIMIT	Precio Limit de	OrderGetDouble, HistoryOrderGetDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	la orden al activarse StopLimit	
ORDER_SL	Nivel Stop Loss	OrderGetDouble, HistoryOrderGetDouble
ORDER_STATE	Estatus de la orden	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_CANCELED	Orden retirada por el cliente	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_EXPIRED	Orden retirada por expirarse el plazo	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_FILLED	Orden ejecutada totalmente	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PARTIAL	Orden ejecutada parcialmente	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PLACED	Orden aceptada	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REJECTED	Orden rechazada	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_ADD	Orden en el estado de	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	registro (colocación en el sistema de trading)	
ORDER_STATE_REQUEST_CANCEL	Orden en el estado de eliminación (eliminación del sistema de trading)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_MODIFY	Orden en el estado de modificación (cambio de parámetros)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_STARTED	Orden verificada pero aún sin aceptar por el corredor	OrderGetInteger, HistoryOrderGetInteger
ORDER_SYMBOL	Símbolo de la orden	OrderGetString, HistoryOrderGetString
ORDER_TIME_DAY	Orden estará vigente sólo en el transcurso del día	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	comercial corriente	
ORDER_TIME_DONE	Hora de ejecución o cancelación de la orden	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE_MSC	Tiempo de ejecución / retirada de la orden en milisegundos desde 01.01.1970	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_EXPIRATION	Plazo de expiración de la orden	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_GTC	Orden estará en la cola hasta que se retire	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP	Hora de establecimiento de la orden	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP_MSC	Tiempo de colocación de la orden para la	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	ejecución en milisegundos desde 01.01.1970	
ORDER_TIME_SPECIFIED	Orden estará vigente hasta que se expire el plazo de vigencia	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED_DAY	Orden se mantiene activa hasta las 23:59:59 del día especificado. Si está hora no cae en ninguna sesión de trading, la expiración llega a la hora de trading más cercana.	OrderGetInteger, HistoryOrderGetInteger
ORDER_TP	Nivel Take Profit	OrderGetDouble, HistoryOrderGetDouble
ORDER_TYPE	Tipo de la orden	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
ORDER_TYPE_BUY	Orden de mercado para la compra	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_LIMIT	Orden pendiente Buy Limit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP	Orden pendiente Buy Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP_LIMIT	Al alcanzar el precio de la orden se coloca la orden pendiente Buy Limit por el precio StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_FILLING	Tipo de ejecución según el resto	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL	Orden de mercado para la venta	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_LIMIT	Orden pendiente Sell Limit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP	Orden pendient	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	e Sell Stop	
ORDER_TYPE_SELL_STOP_LIMIT	Al alcanzar el precio de la orden se coloca la orden pendiente Sell Limit por el precio StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_TIME	Tiempo de vida de la orden	OrderGetInteger, HistoryOrderGetInteger
ORDER_VOLUME_CURRENT	Volumen actual de la orden	OrderGetDouble, HistoryOrderGetDouble
ORDER_VOLUME_INITIAL	Volumen inicial de la orden	OrderGetDouble, HistoryOrderGetDouble
PERIOD_CURRENT	Período corriente	Períodos de gráficos
PERIOD_D1	1 día	Períodos de gráficos
PERIOD_H1	1 hora	Períodos de gráficos
PERIOD_H12	12 horas	Períodos de gráficos
PERIOD_H2	2 horas	Períodos de gráficos
PERIOD_H3	3 horas	Períodos de gráficos
PERIOD_H4	4 horas	Períodos de gráficos
PERIOD_H6	6 horas	Períodos de gráficos
PERIOD_H8	8 horas	Períodos de gráficos

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
PERIOD_M1	1 minuto	Períodos de gráficos
PERIOD_M10	10 minutos	Períodos de gráficos
PERIOD_M12	12 minutos	Períodos de gráficos
PERIOD_M15	15 minutos	Períodos de gráficos
PERIOD_M2	2 minutos	Períodos de gráficos
PERIOD_M20	20 minutos	Períodos de gráficos
PERIOD_M3	3 minutos	Períodos de gráficos
PERIOD_M30	30 minutos	Períodos de gráficos
PERIOD_M4	4 minutos	Períodos de gráficos
PERIOD_M5	5 minutos	Períodos de gráficos
PERIOD_M6	6 minutos	Períodos de gráficos
PERIOD_MN1	1 mes	Períodos de gráficos
PERIOD_W1	1 semana	Períodos de gráficos
PLOT_ARROW	Código de flecha para el estilo DRAW_ARROW	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_ARROW_SHIFT	Desplazamiento vertical de flechas para el estilo	PlotIndexSetInteger, PlotIndexGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	DRAW_ARROW	
PLOT_COLOR_INDEXES	Número de colores	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_BEGIN	Número de barras iniciales sin dibujar y valores en DataWindow	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_TYPE	Tipo de construcción gráfica	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_EMPTY_VALUE	Valor vacío para una representación gráfica, para la que no hay dibujo	PlotIndexSetDouble
PLOT_LABEL	Nombre de la serie gráfica de indicador para la visualización en DataWindow. Para los estilos gráficos complejos, que	PlotIndexSetString

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	requieren varios búferes de indicador para su visualización, los nombres para cada búfer se puede establecer utilizando ";" como separador. El ejemplo del código se puede encontrar en DRAW_CANDLES	
PLOT_LINE_COLOR	Índice del buffer que contiene el color	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_STYLE	Estilo de la línea de dibujo	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_WIDTH	Grosor de línea de dibujo	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHIFT	Desplazamiento de	PlotIndexSetInteger, PlotIndexGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	representación gráfica del indicador respecto al eje de tiempo en barras	
PLOT_SHOW_DATA	Indica la visualización de valores de construcción en la ventana DataWindow	PlotIndexSetInteger, PlotIndexGetInteger
PLUSDI_LINE	Línea +DI	Líneas de indicadores
POINTER_AUTOMATIC	Puntero a cualquier objeto que ha sido creado automáticamente (sin usar new())	CheckPointer
POINTER_DYNAMIC	Puntero a objeto que ha sido creado por el operador new	CheckPointer

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
POINTER_INVALID	Puntero incorrecto	CheckPointer
POSITION_COMMENT	Comentarios de posición	PositionGetString
POSITION_COMMISSION	Comisión	PositionGetDouble
POSITION_IDENTIFIER	Identificador de posición es un número único que se adjudica a cada una de las posiciones abiertas y no se cambia a lo largo de su existencia. La rotación de posición no cambia su identificador.	PositionGetInteger
POSITION_MAGIC	Magic number para la posición (véase ORDER_MAGIC)	PositionGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
POSITION_PRICE_CURRENT	Precio actual para el símbolo	PositionGetDouble
POSITION_PRICE_OPEN	Precio de posición	PositionGetDouble
POSITION_PROFIT	Beneficio corriente	PositionGetDouble
POSITION_SL	Nivel Stop Loss para una posición abierta	PositionGetDouble
POSITION_SWAP	Swap acumulado	PositionGetDouble
POSITION_SYMBOL	Símbolo de posición abierta	PositionGetString
POSITION_TIME	Hora de apertura de posición	PositionGetInteger
POSITION_TIME_MSC	Tiempo de apertura de la posición en milisegundos desde 01.01.1970	PositionGetInteger
POSITION_TIME_UPDATE	Tiempo de modificación de	PositionGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	la posición en segundos desde 01.01.1970	
POSITION_TIME_UPDATE_MSC	Tiempo de modificación de la posición en milisegundos desde 01.01.1970	PositionGetInteger
POSITION_TP	Nivel Take Profit para una posición abierta	PositionGetDouble
POSITION_TYPE	Tipo de posición	PositionGetInteger
POSITION_TYPE_BUY	Compra	PositionGetInteger
POSITION_TYPE_SELL	Venta	PositionGetInteger
POSITION_VOLUME	Volumen de posición	PositionGetDouble
PRICE_CLOSE	Precio de cierre	Constantes de precio
PRICE_HIGH	Precio máximo en el período	Constantes de precio
PRICE_LOW	Precio mínimo en el período	Constantes de precio

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
PRICE_MEDIAN	Precio mediano, (high+low)/2	Constantes de precio
PRICE_OPEN	Precio de apertura	Constantes de precio
PRICE_TYPICAL	Precio típico, (high+low+close)/3	Constantes de precio
PRICE_WEIGHTED	Precio promedio, (high+low+close+close)/4	Constantes de precio
PROGRAM_EXPERT	Experto	MQLInfoInteger
PROGRAM_INDICATOR	Indicador	MQLInfoInteger
PROGRAM_SCRIPT	Script	MQLInfoInteger
REASON_ACCOUNT	Ha sido activada otra cuenta o ha tenido lugar la reconexión con el servidor comercial debido al cambio de los ajustes de la cuenta	UninitializeReason, OnDeinit

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
REASON_CHARTCHANGE	El símbolo o período del gráfico ha sido modificado	UninitializeReason, OnDeinit
REASON_CHARTCLOSE	El gráfico ha sido cerrado	UninitializeReason, OnDeinit
REASON_CLOSE	El terminal ha sido cerrado	UninitializeReason, OnDeinit
REASON_INITFAILED	Este valor significa que el manejador OnInit() ha devuelto un valor no nulo	UninitializeReason, OnDeinit
REASON_PARAMETERS	Los parámetros de entrada han sido cambiados por el usuario	UninitializeReason, OnDeinit
REASON_PROGRAM	Asesor Experto finalizó su trabajo llamando a la función	UninitializeReason, OnDeinit

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	ExpertRemove()	
REASON_RECOMPILE	El programa ha sido recompilado	UninitializeReason, OnDeinit
REASON_REMOVE	El programa ha sido eliminado del gráfico	UninitializeReason, OnDeinit
REASON_TEMPLATE	Ha sido aplicada la nueva plantilla del gráfico	UninitializeReason, OnDeinit
SATURDAY	Sábado	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SEEK_CUR	Posición actual de un puntero de archivos	FileSeek
SEEK_END	Final del archivo	FileSeek
SEEK_SET	Principio del archivo	FileSeek
SENKOUSPANA_LINE	Línea Senkou Span A	Líneas de indicadores
SENKOUSPANB_LINE	Línea Senkou Span B	Líneas de indicadores

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
SERIES_BARS_COUNT	Número de barras para el símbolo-período en el momento actual	SeriesInfoInteger
SERIES_FIRSTDATE	La primera fecha para el símbolo-período en el momento actual	SeriesInfoInteger
SERIES_LASTBAR_DATE	Información sobre datos históricos del instrumento	SeriesInfoInteger
SERIES_SERVER_FIRSTDATE	La primera fecha en el historial para el símbolo en el servidor independientemente del período	SeriesInfoInteger
SERIES_SYNCHRONIZED	Significa la sincronización de datos para el	SeriesInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	símbolo/período en este momento	
SERIES_TERMINAL_FIRSTDATE	La primera fecha en el historial para el símbolo en el terminal de cliente independientemente del período	SeriesInfoInteger
SHORT_MAX	Valor máximo que puede ser representado por el tipo short	Constantes de tipos numéricos
SHORT_MIN	Valor mínimo que puede ser representado por el tipo short	Constantes de tipos numéricos
SIGNAL_BASE_AUTHOR_LOGIN	Login del autor de la señal	SignalBaseGetString
SIGNAL_BASE_BALANCE	Balance de la cuenta	SignalBaseGetDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
SIGNAL_BASE_BROKER	Nombre del broker (empresa)	SignalBaseGetString
SIGNAL_BASE_BROKER_SERVER	Servidor del broker	SignalBaseGetString
SIGNAL_BASE_CURRENCY	Divisa de la cuenta de la señal	SignalBaseGetString
SIGNAL_BASE_DATE_PUBLISHED	Fecha de publicación de la señal (cuando se ha hecho disponible)	SignalBaseGetInteger
SIGNAL_BASE_DATE_STARTED	Fecha del inicio del monitoreo de la señal	SignalBaseGetInteger
SIGNAL_BASE_EQUITY	Fondos en la cuenta	SignalBaseGetDouble
SIGNAL_BASE_GAIN	Crecimiento de la cuenta en por cientos	SignalBaseGetDouble
SIGNAL_BASE_ID	ID de la señal	SignalBaseGetInteger
SIGNAL_BASE_LEVERAGE	Apalancamiento de la cuenta	SignalBaseGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	de trading	
SIGNAL_BASE_MAX_DRAWDOWN	Reducción máxima	SignalBaseGetDouble
SIGNAL_BASE_NAME	Nombre de la señal	SignalBaseGetString
SIGNAL_BASE_PIPS	Resultado del trading en pips	SignalBaseGetInteger
SIGNAL_BASE_PRICE	Precio de la suscripción a la señal	SignalBaseGetDouble
SIGNAL_BASE_RATING	Posición en el ranking de las señales	SignalBaseGetInteger
SIGNAL_BASE_ROI	Valor ROI (Return on Investment) de la señal en %	SignalBaseGetDouble
SIGNAL_BASE_SUBSCRIBERS	Número de suscriptores	SignalBaseGetInteger
SIGNAL_BASE_TRADE_MODE	Tipo de la cuenta (0-real, 1-demo, 2-de concurso)	SignalBaseGetInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
SIGNAL_BASE_TRADES	Número de trades	SignalBaseGetInteger
SIGNAL_INFO_CONFIRMATIONS_DISABLED	Bandera del permiso para la sincronización sin mostrar el diálogo de confirmación	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_COPY_SLTP	Bandera del copiado de Stop Loss y Take Profit	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_DEPOSIT_PERCENT	Limitación del depósito (en %)	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_EQUITY_LIMIT	Porcentaje para la conversión del volumen de la transacción	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_ID	id de la señal, r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_NAME	Nombre de la señal, r/o	SignalInfoGetString

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
SIGNAL_INFO_SLIPPAGE	Valor del deslizamiento con el que se coloca la orden de mercado durante la sincronización y copiado de las transacciones	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_SUBSCRIPTION_ENABLED	Bandera del permiso para el copiado de las transacciones según la suscripción	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_TERMS_AGREE	Bandera de aceptación de las condiciones del uso del servicio "Señales", r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_VOLUME_PERCENT	Valor de limitación de fondos para la señal, r/o	SignalInfoGetDouble, SignalInfoSetDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
SIGNAL_LINE	Línea de señales	Líneas de indicadores
STAT_BALANCE_DD	Reducción máxima del balance en dinero. En el proceso de trading el balance puede sufrir varias reducciones, se coge el valor máximo.	TesterStatistics
STAT_BALANCE_DD_RELATIVE	Reducción del balance en dinero que ha sido detectada en el momento de la reducción máxima del balance en por cientos (STAT_BALANCE_DDREL_PERCENT).	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
STAT_BALANCE_DDREL_PERCENT	Reducción máxima del balance en por cientos. En el proceso de trading el balance puede sufrir varias reducciones, para cada una de ellas se calcula el valor relativo de reducción en por cientos. Se devuelve el valor máximo	TesterStatistics
STAT_BALANCEDD_PERCENT	Reducción del balance en por cientos que ha sido detectada en el momento de la reducción máxima	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	del balance en dinero (STAT_BALANCE_DD).	
STAT_BALANCEMIN	Valor mínimo del balance	TesterStatistics
STAT_CONLOSSMAX	Pérdida máxima en una secuencia de transacciones de pérdidas. Su valor es inferior o igual a cero	TesterStatistics
STAT_CONLOSSMAX_TRADES	Número de transacciones que han formado STAT_CONLOSSMAX (pérdida máxima en una secuencia de transacciones de pérdidas)	TesterStatistics
STAT_CONPROFITMAX	Beneficio máximo	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	en una secuencia de transacciones rentables. Su valor es superior o igual a cero	
STAT_CONPROFITMAX_TRADES	Número de transacciones que han formado STAT_CONPROFITMAX (beneficio máximo en una secuencia de transacciones rentables)	TesterStatistics
STAT_CUSTOM_ONTESTER	Valor del criterio de optimización de usuario, calculado y devuelto por la función OnTester()	TesterStatistics
STAT_DEALS	Número de transacc	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	iones realizadas	
STAT_EQUITY_DD	Reducción máxima de equidad en dinero. En el proceso de trading la equidad puede sufrir varias reducciones, se coge el valor máximo.	TesterStatistics
STAT_EQUITY_DD_RELATIVE	Reducción de equidad en dinero que ha sido detectada en el momento de la reducción máxima de equidad en por cientos (STAT_EQUITY_DDREL_PERCENT).	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
STAT_EQUITY_DDREL_PERCENT	Reducción máxima de equidad en por cientos. En el proceso de trading la equidad puede sufrir varias reducciones, para cada una de ellas se calcula el valor relativo de reducción en por cientos. Se devuelve el valor máximo	TesterStatistics
STAT_EQUITYDD_PERCENT	Reducción de equidad en por cientos que ha sido detectada en el momento de la reducción máxima	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	de equidad en dinero (STAT_EQUITY_DD).	
STAT_EQUITYMIN	Valor mínimo de equidad	TesterStatistics
STAT_EXPECTED_PAYOFF	Beneficio esperado	TesterStatistics
STAT_GROSS_LOSS	Pérdidas brutas, importe de todas las transacciones de pérdidas (con resultados negativos). Su valor es inferior o igual a cero	TesterStatistics
STAT_GROSS_PROFIT	Beneficio bruto, importe de todas las transacciones rentables (con resultados positivos). Su	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	valor es superior o igual a cero	
STAT_INITIAL_DEPOSIT	Valor del depósito inicial	TesterStatistics
STAT_LONG_TRADES	Trades largos	TesterStatistics
STAT_LOSS_TRADES	Trades irrentables	TesterStatistics
STAT_LOSSTRADES_AVGCON	Longitud media de una serie irrentable de trades	TesterStatistics
STAT_MAX_CONLOSS_TRADES	Número de transacciones en la serie más larga de transacciones de pérdidas STAT_MAX_CONLOSSES	TesterStatistics
STAT_MAX_CONLOSSES	Pérdidas totales en la serie más larga de transacciones de pérdidas	TesterStatistics
STAT_MAX_CONPROFIT_TRADES	Numero de	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	transacciones en la serie más larga de transacciones rentables STAT_MAX_CONWINS	
STAT_MAX_CONWINS	Beneficio total en la serie más larga de transacciones rentables	TesterStatistics
STAT_MAX_LOSSTRADE	Pérdida máxima - el valor mínimo entre todas las transacciones de pérdidas. Su valor es inferior o igual a cero	TesterStatistics
STAT_MAX_PROFITTRADE	Beneficio máximo - el valor máximo entre todas las transacciones rentable	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	s. Su valor es superior o igual a cero	
STAT_MIN_MARGINLEVEL	Valor mínimo alcanzado del nivel de margen	TesterStatistics
STAT_PROFIT	Beneficio neto tras la simulación, suma de STAT_GROSS_PROFIT y STAT_GROSS_LOSS (STAT_GROSS_LOSS siempre es menos o igual a cero)	TesterStatistics
STAT_PROFIT_FACTOR	Factor de beneficio - relación entre STAT_GROSS_PROFIT/STAT_GROSS_LOSS. Si STAT_GROSS_LOSS=0,	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	entonces el Factor de Beneficio obtiene el valor DBL_MAX	
STAT_PROFIT_LONGTRADES	Trades largos rentables	TesterStatistics
STAT_PROFIT_SHORTTRADES	Trades cortos rentables	TesterStatistics
STAT_PROFIT_TRADES	Trades rentables	TesterStatistics
STAT_PROFITTRADES_AVGCON	Longitud media de una serie rentable de trades	TesterStatistics
STAT_RECOVERY_FACTOR	Factor de recuperación - relación entre STAT_PROFIT/STAT_BALANCE_DD	TesterStatistics
STAT_SHARPE_RATIO	Ratio de Sharpe	TesterStatistics
STAT_SHORT_TRADES	Trades cortos	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
STAT_TRADES	Número de trades	TesterStatistics
STAT_WITHDRAWAL	Fondos retirados de la cuenta	TesterStatistics
STO_CLOSECLOSE	Calcular basándose en los precios Close/Close	Constantes de precio
STO_LOWHIGH	Calcular basándose en los precios Low/High	Constantes de precio
STYLE_DASH	Polilínea	Estilos de dibujo
STYLE_DASHDOT	Línea de punto y raya	Estilos de dibujo
STYLE_DASHDOTDOT	Línea de dos puntos y raya	Estilos de dibujo
STYLE_DOT	Línea de puntos	Estilos de dibujo
STYLE_SOLID	Línea continua	Estilos de dibujo
SUNDAY	Domingo	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SYMBOL_ASK	Ask - mejor oferta de compra	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
SYMBOL_ASKHIGH	Ask máximo del día	SymbolInfoDouble
SYMBOL_ASKLOW	Ask mínimo del día	SymbolInfoDouble
SYMBOL_BANK	Fuente de cotización actual	SymbolInfoString
SYMBOL_BASIS	Nombre del activo base para el símbolo	SymbolInfoString
SYMBOL_BID	Bid - mejor oferta de venta	SymbolInfoDouble
SYMBOL_BIDHIGH	Bid máximo del día	SymbolInfoDouble
SYMBOL_BIDLOW	Bid mínimo del día	SymbolInfoDouble
SYMBOL_CALC_MODE_CFD	CFD mode - cálculo de beneficio y margen para CFD	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDINDEX	CFD index mode - cálculo de beneficio y margen	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	para CFD por índices	
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD Leverage mode - cálculo de beneficio y margen para CFD en caso del trading con apalancamiento financiero	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES	Futures mode - cálculo de beneficio y margen para tradear con los contratos de futuros en la bolsa	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS Futures mode - cálculo de beneficio y margen para tradear con los	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	contratos de futuros en FORTS.	
SYMBOL_CALC_MODE_EXCH_STOCKS	Exchange mode - cálculo de beneficio y margen para tradear con los valores en la bolsa	SymbolInfoInteger
SYMBOL_CALC_MODE_FOREX	Forex mode - cálculo de beneficio y margen para Forex	SymbolInfoInteger
SYMBOL_CALC_MODE_FUTURES	Futures mode - cálculo de beneficio y margen para futuros	SymbolInfoInteger
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - el símbolo se usa como un activo no tradeabl	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	e en la cuenta comercial.	
SYMBOL_CURRENCY_BASE	Divisa básica del instrumento	SymbolInfoString
SYMBOL_CURRENCY_MARGIN	Divisa de margen	SymbolInfoString
SYMBOL_CURRENCY_PROFIT	Divisa de beneficio	SymbolInfoString
SYMBOL_DESCRIPTION	Descripción literal del símbolo	SymbolInfoString
SYMBOL_DIGITS	Número de dígitos después del punto decimal	SymbolInfoInteger
SYMBOL_EXPIRATION_DAY	La orden está vigente hasta que se termine el día	SymbolInfoInteger
SYMBOL_EXPIRATION_GTC	La orden está vigente sin restricción de tiempo hasta su	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	explícita cancelación	
SYMBOL_EXPIRATION_MODE	Banderas de los modos de expiración de la orden permitidos	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED	El plazo de vencimiento se indica en la orden	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED_DAY	El día de vencimiento se especifica en la orden	SymbolInfoInteger
SYMBOL_EXPIRATION_TIME	Fecha final de licitaciones por herramienta (normalmente se utiliza para los futuros)	SymbolInfoInteger
SYMBOL_FILLING_FOK	Esta política de ejecución significa que la orden	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	puede ser ejecutada exclusivamente en el volumen indicado. Si en este momento en el mercado no hay volumen necesario del instrumento financiero requerido, esta orden no se ejecutará. El volumen necesario puede ser cubierto por varias ofertas disponibles en este momento en el mercado.	
SYMBOL_FILLING_IOC	En este caso el trader acepta	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	realizar la transacción en el volumen máximo disponible en el mercado dentro del margen especificado en la orden. Si la ejecución completa no es posible, la orden será ejecutada en el volumen disponible, y el resto del volumen no cubierto será cancelado. La posibilidad de ejecución IOC de las órdenes se determina en el servidor comercial.	

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
SYMBOL_FILLING_MODE	Banderas de los modos de relleno de la orden permitidos	SymbolInfoInteger
SYMBOL_ISIN	Nombre del símbolo comercial en el sistema de los códigos internacionales para identificación de valores — ISIN (International Securities Identification Number). El código internacional de identificación de un valores un código de 12 dígitos que contiene letras y cifras y que	SymbolInfoString

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	identifica de forma unívoca a un valor mobiliario a nivel internacional. La presencia de esta propiedad del símbolo se determina en el lado del servidor comercial.	
SYMBOL_LAST	Precio de la última transacción	SymbolInfoDouble
SYMBOL_LASTHIGH	Last máximo del día	SymbolInfoDouble
SYMBOL_LASTLOW	Last mínimo del día	SymbolInfoDouble
SYMBOL_MARGIN_INITIAL	Margen inicial (inicializador) significa el monto de fondos asegurados necesari	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	os en la moneda de margen para abrir posiciones en el volumen de un lote. Se utiliza para verificar los fondos del cliente al entrar en el merado.	
SYMBOL_MARGIN_MAINTENANCE	Margen de mantenimiento del instrumento. En caso de establecerlo - indica el monto del margen en la moneda de margen del instrumento que se retiene de un lote. Se utiliza	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	para verificar los fondos del cliente cuando se cambia el estado de la cuenta del cliente. Si el margen de mantenimiento es igual a 0, se utiliza el margen inicial.	
SYMBOL_OPTION_MODE	Tipo de opción	SymbolInfoInteger
SYMBOL_OPTION_MODE_AMERICAN	Tipo europeo de la opción - puede ser amortizado sólo en la fecha especificada (fecha de vencimiento del plazo, fecha de ejecució	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	n, fecha de reembolso)	
SYMBOL_OPTION_MODE_EUROPEAN	Tipo americano de la opción - puede ser amortizado en cualquier fecha antes del vencimiento de la opción. Para este tipo se establece un período durante el cual el comprador puede ejecutar esta opción	SymbolInfoInteger
SYMBOL_OPTION_RIGHT	Derecho de la opción (Call/Put)	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_CALL	La opción que concede el derecho a comprar	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	el activo por el precio fijo	
SYMBOL_OPTION_RIGHT_PUT	La opción que concede el derecho a vender el activo por el precio fijo	SymbolInfoInteger
SYMBOL_OPTION_STRIKE	Precio de ejecución de la opción. Es el precio por el que el comprador de la opción puede comprar (si la opción es Call) o vender (si la opción es Put) un activo base, y el vendedor de la opción, consecuentemente, está	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	obligado a vender o comprar la cantidad correspondiente del activo base	
SYMBOL_ORDER_LIMIT	Están permitidas las órdenes limitadas (Buy Limit y Sell Limit)	SymbolInfoInteger
SYMBOL_ORDER_MARKET	Están permitidas las órdenes de mercado (Buy y Sell en el mercado sin especificar el precio de transacción)	SymbolInfoInteger
SYMBOL_ORDER_MODE	Banderas de los tipos de la orden permitidos	SymbolInfoInteger
SYMBOL_ORDER_SL	Está permitida la	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	colocación de Stop Loss	
SYMBOL_ORDER_STOP	Están permitidas las órdenes Stop (Buy Stop y Sell Stop)	SymbolInfoInteger
SYMBOL_ORDER_STOP_LIMIT	Están permitidas las órdenes Stop Limit (Buy Stop Limit y Sell Stop Limit)	SymbolInfoInteger
SYMBOL_ORDER_TP	Está permitida la colocación de Take Profit	SymbolInfoInteger
SYMBOL_PATH	Ruta en el árbol de símbolos	SymbolInfoString
SYMBOL_POINT	Valor de un punto	SymbolInfoDouble
SYMBOL_SELECT	Indica que el símbolo ha sido seleccionado en	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	Market Watch	
SYMBOL_SESSION_AW	Precio medio ponderado de la sesión	SymbolInfoDouble
SYMBOL_SESSION_BUY_ORDERS	Número total de órdenes de compra en el momento actual	SymbolInfoInteger
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Volumen total de órdenes de compra en el momento actual	SymbolInfoDouble
SYMBOL_SESSION_CLOSE	Precio de cierre de la sesión	SymbolInfoDouble
SYMBOL_SESSION_DEALS	Número de transacciones en la sesión actual	SymbolInfoInteger
SYMBOL_SESSION_INTEREST	Volumen total de posiciones abiertas	SymbolInfoDouble
SYMBOL_SESSION_OPEN	Precio de apertura de la sesión	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
SYMBOL_SESSION_PRICE_LIMIT_MAX	Precio máximo aceptable para la sesión	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MIN	Precio mínimo aceptable para la sesión	SymbolInfoDouble
SYMBOL_SESSION_PRICE_SETTLEMENT	Precio de entrega para la sesión actual	SymbolInfoDouble
SYMBOL_SESSION_SELL_ORDERS	Número total de órdenes de venta en el momento actual	SymbolInfoInteger
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Volumen total de órdenes de venta en el momento actual	SymbolInfoDouble
SYMBOL_SESSION_TURNOVER	Circulación total durante la sesión actual	SymbolInfoDouble
SYMBOL_SESSION_VOLUME	Volumen total de transacciones de la sesión actual	SymbolInfoDouble
SYMBOL_SPREAD	Tamaño de	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	spread en puntos	
SYMBOL_SPREAD_FLOAT	Indicio del spread flotante	SymbolInfoInteger
SYMBOL_START_TIME	Fecha de inicio de licitaciones por instrumento (normalmente se utiliza para los futuros)	SymbolInfoInteger
SYMBOL_SWAP_LONG	Valor de swap long	SymbolInfoDouble
SYMBOL_SWAP_MODE	Modelo para calcular swap	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Swaps se calculan en dinero en divisa del depósito del cliente	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Swaps se calculan en dinero en divisa	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	marginal del símbolo	
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Swaps se calculan en dinero en divisa base del símbolo	SymbolInfoInteger
SYMBOL_SWAP_MODE_DISABLED	No hay swaps	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Swaps se calculan en por cientos anuales del precio del instrumento para el momento de cálculo del swap (régimen bancarios de 360 días al año)	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_OPEN	Swaps se calculan en por cientos anuales del precio de	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	apertura de la posición para el símbolo (régimen bancarios de 360 días al año)	
SYMBOL_SWAP_MODE_POINTS	Swaps se calculan en puntos	SymbolInfoInteger
SYMBOL_SWAP_MODE_REOPEN_BID	Swaps se calculan por reapertura de posiciones. Al final de la sesión de trading la posición se cierra forzosamente. Al día siguiente la posición vuelve a abrirse por el precio actual Bid +/- el número de	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	puntos especificado (en los parámetros SYMBOL_SWAP_LONG y SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Swaps se calculan por reapertura de posiciones. Al final de la sesión de trading la posición se cierra forzosamente. Al día siguiente la posición vuelve a abrirse por el precio de cierre +/- el número de puntos especificado (en los parámetros	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	SYMBOL_SWAP_LONG y SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_ROLLOVER3DAYS	Día de la semana para cargar la refinanciación del swap de 3 días	SymbolInfoInteger
SYMBOL_SWAP_SHORT	Valor de swap short	SymbolInfoDouble
SYMBOL_TICKS_BOOKDEPTH	Cantidad máxima de las solicitudes mostradas en la profundidad. Para los instrumentos sin cola de solicitudes, el valor es 0	SymbolInfoInteger
SYMBOL_TIME	Hora de la última cotización	SymbolInfoInteger
SYMBOL_TRADE_CALC_MODE	Modo de calcular el coste del contrato	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
SYMBOL_TRADE_CONTRACT_SIZE	Tamaño del contrato comercial	SymbolInfoDouble
SYMBOL_TRADE_EXECUTION_EXCHANGE	Ejecución por Bolsa	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_INSTANT	Ejecución Instantánea	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_MARKET	Ejecución de órdenes por Mercado	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_REQUEST	Ejecución por Pedido	SymbolInfoInteger
SYMBOL_TRADE_EXEMODE	Modo de ejecución de transacciones	SymbolInfoInteger
SYMBOL_TRADE_FREEZE_LEVEL	Distancia de congelamiento de operaciones comerciales (en puntos)	SymbolInfoInteger
SYMBOL_TRADE_MODE	Tipo de ejecución de órdenes	SymbolInfoInteger
SYMBOL_TRADE_MODE_CLOSEONLY	Permitidas sólo operaciones de	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	cierre de posiciones	
SYMBOL_TRADE_MODE_DISABLED	Trading por el símbolo prohibido	SymbolInfoInteger
SYMBOL_TRADE_MODE_FULL	Sin restricciones de las operaciones comerciales	SymbolInfoInteger
SYMBOL_TRADE_MODE_LONGONLY	Sólo compras	SymbolInfoInteger
SYMBOL_TRADE_MODE_SHORTONLY	Sólo ventas	SymbolInfoInteger
SYMBOL_TRADE_STOPS_LEVEL	Margen mínimo en puntos del actual precio de cierre para la colocación de las órdenes Stop	SymbolInfoInteger
SYMBOL_TRADE_TICK_SIZE	Mínimo cambio de precio	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE	Valor SYMBOL_TRADE_TICK_VALUE_PROFIT	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
SYMBOL_TRADE_TICK_VALUE_LOSS	Precio calculado del tick para la posición no rentable	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_PROFIT	Precio calculado del tick para la posición rentable	SymbolInfoDouble
SYMBOL_VOLUME	Volume - volumen en la última transacción	SymbolInfoInteger
SYMBOL_VOLUME_LIMIT	Volumen total máximo permitido de la posición abierta y órdenes pendientes (independientemente de la dirección) para un símbolo	SymbolInfoDouble
SYMBOL_VOLUME_MAX	Volumen máximo para celebrar una	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	transacción	
SYMBOL_VOLUME_MIN	Volumen mínimo para celebrar una transacción	SymbolInfoDouble
SYMBOL_VOLUME_STEP	Paso mínimo del cambio de volumen para celebrar una transacción	SymbolInfoDouble
SYMBOL_VOLUMEHIGH	Volumen máximo del día	SymbolInfoInteger
SYMBOL_VOLUMELOW	Volumen mínimo del día	SymbolInfoInteger
TENKANSEN_LINE	Línea Tenkan-sen	Líneas de indicadores
TERMINAL_BUILD	Número de la build del terminal	TerminalInfoInteger
TERMINAL_CODEPAGE	Número de la página de código del idioma instalado en el terminal	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	de cliente	
TERMINAL_COMMONDATA_PATH	Carpeta general de todos los terminales de cliente instalados en el ordenador	TerminalInfoString
TERMINAL_COMMUNITY_ACCOUNT	Bandera de la presencia de los datos de autorización de MQL5.community en el terminal	TerminalInfoInteger
TERMINAL_COMMUNITY_BALANCE	Balance del usuario en MQL5.community	TerminalInfoDouble
TERMINAL_COMMUNITY_CONNECTION	Conexión a MQL5.community	TerminalInfoInteger
TERMINAL_COMPANY	Nombre de la empresa	TerminalInfoString
TERMINAL_CONNECTED	Conexión al servidor comercial	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
TERMINAL_CPU_CORES	Número de procesadores en el sistema	TerminalInfoInteger
TERMINAL_DATA_PATH	Carpeta donde se almacenan los datos del terminal	TerminalInfoString
TERMINAL_DISK_SPACE	Tamaño de la memoria libre en el disco para la carpeta MQL5\Files del terminal (agente), en MB	TerminalInfoInteger
TERMINAL_DLLS_ALLOWED	Permiso de usar DLL	TerminalInfoInteger
TERMINAL_EMAIL_ENABLED	Permiso de enviar correo electrónico usando el servidor SMTP y nombre de usuario especificados en las configur	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	aciones del terminal	
TERMINAL_FTP_ENABLED	Permiso de enviar informes a través de FTP a un servidor indicado para una cuenta comercial especificada en las configuraciones del terminal	TerminalInfoInteger
TERMINAL_LANGUAGE	Idioma del terminal	TerminalInfoString
TERMINAL_MAXBARS	Número máximo de barras en el gráfico	TerminalInfoInteger
TERMINAL_MEMORY_AVAILABLE	Tamaño de la memoria libre del proceso del terminal (agente), en MB	TerminalInfoInteger
TERMINAL_MEMORY_PHYSICAL	Tamaño de la memoria	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	física en el sistema, en MB	
TERMINAL_MEMORY_TOTAL	Tamaño de la memoria disponible para el proceso del terminal (agente), en MB	TerminalInfoInteger
TERMINAL_MEMORY_USED	Tamaño de la memoria usada por el terminal (agente), en MB	TerminalInfoInteger
TERMINAL_MQID	Bandera de presencia de MetaQuotes ID para el envío de las notificaciones Push	TerminalInfoInteger
TERMINAL_NAME	Nombre del terminal	TerminalInfoString
TERMINAL_NOTIFICATIONS_ENABLED	Permiso para enviar las notificaciones al smartphone	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
TERMINAL_OPENCL_SUPPORT	Versión de OpenCL soportada en el formato 0x00010002 = 1.2. "0" significa que OpenCL no se soporta	TerminalInfoInteger
TERMINAL_PATH	Carpeta de la que se inicia el terminal	TerminalInfoString
TERMINAL_PING_LAST	Der letzte bekannte Wert des Pings zum Handelsserver in Mikrosekunden. Eine Mikrosekunde ist eine millionstel Sekunde	TerminalInfoInteger
TERMINAL_SCREEN_DPI	La capacidad de resolución a la hora de mostrar información en la	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	pantalla se mide por la cantidad de puntos por pulgada lineal de la superficie (DPI)	
TERMINAL_TRADE_ALLOWED	Permiso de hacer comercio	TerminalInfoInteger
TERMINAL_X64	Indicación "terminal de 64 bits"	TerminalInfoInteger
THURSDAY	Jueves	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TRADE_ACTION_DEAL	Colocar una orden comercial de conclusión inmediata de una transacción según los parámetros especificados (colocar una orden de	MqlTradeRequest

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	mercado)	
TRADE_ACTION_MODIFY	Modificar los parámetros de una orden comercial colocada anteriormente	MqlTradeRequest
TRADE_ACTION_PENDING	Colocar una orden comercial para la conclusión de una transacción bajo unas condiciones especificadas (orden pendiente)	MqlTradeRequest
TRADE_ACTION_REMOVE	Eliminar una orden pendiente colocada anteriormente	MqlTradeRequest
TRADE_ACTION_SLTP	Modificar los valores Stop Loss y	MqlTradeRequest

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	Take Profit de una posición abierta	
TRADE_RETCODE_CANCEL	Solicitud cancelada por el agente	MqlTradeResult
TRADE_RETCODE_CLIENT_DISABLES_AT	Autotrading está prohibido por el terminal de cliente	MqlTradeResult
TRADE_RETCODE_CONNECTION	No hay conexión con el servidor de comercio	MqlTradeResult
TRADE_RETCODE_DONE	Solicitud ejecutada	MqlTradeResult
TRADE_RETCODE_DONE_PARTIAL	Solicitud ejecutada parcialmente	MqlTradeResult
TRADE_RETCODE_ERROR	Error al procesar la solicitud	MqlTradeResult
TRADE_RETCODE_FROZEN	Orden o posición están congeladas	MqlTradeResult
TRADE_RETCODE_INVALID	Solicitud no válida	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
TRADE_RETCODE_INVALID_EXPIRATION	Fecha de expiración no válida de la orden en la solicitud	MqlTradeResult
TRADE_RETCODE_INVALID_FILL	Está especificado el tipo de ejecución de orden no válido	MqlTradeResult
TRADE_RETCODE_INVALID_ORDER	Tipo de orden inválido o prohibido	MqlTradeResult
TRADE_RETCODE_INVALID_PRICE	Precio en la solicitud no válido	MqlTradeResult
TRADE_RETCODE_INVALID_STOPS	Stops en la solicitud no válido	MqlTradeResult
TRADE_RETCODE_INVALID_VOLUME	Volumen en la solicitud no válido	MqlTradeResult
TRADE_RETCODE_LIMIT_ORDERS	Alcanzado el límite del número de órdenes	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	pendientes	
TRADE_RETCODE_LIMIT_VOLUME	Alcanzado el límite del volumen de órdenes y posiciones para este símbolo	MqlTradeResult
TRADE_RETCODE_LOCKED	Solicitud está bloqueada para procesar	MqlTradeResult
TRADE_RETCODE_MARKET_CLOSED	Mercado está cerrado	MqlTradeResult
TRADE_RETCODE_NO_CHANGES	Sin cambios en la solicitud	MqlTradeResult
TRADE_RETCODE_NO_MONEY	Falta de medios monetarios para cumplir la solicitud	MqlTradeResult
TRADE_RETCODE_ONLY_REAL	Operación permitida únicamente para las cuentas reales	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
TRADE_RETCODE_ORDER_CHANGED	Estado de la orden se ha cambiado	MqlTradeResult
TRADE_RETCODE_PLACED	Orden colocada	MqlTradeResult
TRADE_RETCODE_POSITION_CLOSED	Posición con el POSITION_IDENTIFIER especificado ya está cerrada	MqlTradeResult
TRADE_RETCODE_PRICE_CHANGED	Precios se han cambiado	MqlTradeResult
TRADE_RETCODE_PRICE_OFF	Faltan las cotizaciones para procesar la solicitud	MqlTradeResult
TRADE_RETCODE_REJECT	Solicitud rechazada	MqlTradeResult
TRADE_RETCODE_REQUOTE	Recuota	MqlTradeResult
TRADE_RETCODE_SERVER_DISABLES_AT	Autotrading está prohibido por el servidor	MqlTradeResult
TRADE_RETCODE_TIMEOUT	Solicitud cancelada al expirar el plazo	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
TRADE_RETCODE_TOO_MANY_REQUESTS	Solicitudes muy frecuentes	MqlTradeResult
TRADE_RETCODE_TRADE_DISABLED	Transacciones comerciales están prohibidas	MqlTradeResult
TRADE_TRANSACTION_DEAL_ADD	Agregación de una transacción (deal) al historial. Se hace como resultado de ejecución de la orden o realización de la operación con el balance de la cuenta.	MqlTradeTransaction
TRADE_TRANSACTION_DEAL_DELETE	Eliminación de una transacción (deal) del historial. Puede pasar que una transacción	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	(deal) ejecutada antes se elimine en el servidor. Por ejemplo, la transacción (deal) fue eliminada en el sistema externo de trading (bolsa) a donde había sido pasada por el broker.	
TRADE_TRANSACTION_DEAL_UPDATE	Modificación de una transacción (deal) en el historial. Puede pasar que una transacción (deal) ejecutada antes se cambie en el servidor	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	. Por ejemplo, la transacción (deal) fue modificada en el sistema externo de trading (bolsa) a donde había sido pasada por el broker.	
TRADE_TRANSACTION_HISTORY_ADD	Agregación de una orden al historial como resultado de su ejecución o cancelación.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_DELETE	Eliminación de una orden del historial de órdenes. Este tipo está previsto para ampliar la	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	funcionalidad en la parte del servidor.	
TRADE_TRANSACTION_HISTORY_UPDATE	Modificación de una orden que se encuentra en el historial de órdenes. Este tipo está previsto para ampliar la funcionalidad en la parte del servidor.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_ADD	Agregación de una nueva orden abierta.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_DELETE	Eliminación de la orden de la lista de órdenes abiertas. Una orden puede ser eliminad	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	a de las abiertas como resultado de colocación de la solicitud correspondiente, o bien una vez ejecutada (llena) y pasada al historial.	
TRADE_TRANSACTION_ORDER_UPDATE	Modificación de orden abierta. A estas modificaciones les corresponden no sólo los cambios explícitos por parte del terminal de cliente o servidor de trading, sino también alteraciones de su estado durante la	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	colocación (por ejemplo, paso del estado ORDER_STATE_STARTED a ORDER_STATE_PLACED o de ORDER_STATE_PLACED a ORDER_STATE_PARTIAL etc.).	
TRADE_TRANSACTION_POSITION	Modificación de la posición que no está relacionada con la ejecución de la transacción (deal). Este tipo de transacción (transaction) quiere decir que la posición ha sido modificada en la	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	parte del servidor de trading. La posición puede sufrir el cambio del volumen, precio de apertura, así como de los niveles Stop Loss y Take Profit. La información sobre los cambios se envía en la estructura MqlTradeTransaction usando el manejador OnTradeTransaction. La modificación de una posición (agregación, cambio	

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	o eliminación) como resultado de ejecución de la transacción (deal) no supone la aparición tras sí la transacción (transaction) TRADE_TRANSACTION_POSITION.	
TRADE_TRANSACTION_REQUEST	Aviso de que la solicitud comercial ha sido procesada por el servidor, y el resultado de su procesamiento ha sido recibido. Para las transacciones de este tipo en la estructu	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	ra MqlTradeTransaction hay que analizar sólo un campo - type (tipo de transacción). Para obtener la información adicional hay que analizar el segundo y el tercer parámetro de la función OnTradeTransaction (request y result).	
TUESDAY	Martes	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TYPE_BOOL	bool	MqlParam
TYPE_CHAR	char	MqlParam
TYPE_COLOR	color	MqlParam
TYPE_DATETIME	datetime	MqlParam
TYPE_DOUBLE	double	MqlParam
TYPE_FLOAT	float	MqlParam

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
TYPE_INT	int	MqlParam
TYPE_LONG	long	MqlParam
TYPE_SHORT	short	MqlParam
TYPE_STRING	string	MqlParam
TYPE_UCHAR	uchar	MqlParam
TYPE_UINT	uint	MqlParam
TYPE_ULONG	ulong	MqlParam
TYPE_USHORT	ushort	MqlParam
UCHAR_MAX	Valor máximo que puede ser representado por el tipo uchar	Constantes de tipos numéricos
UINT_MAX	Valor máximo que puede ser representado por el tipo uint	Constantes de tipos numéricos
ULONG_MAX	Valor máximo que puede ser representado por el tipo ulong	Constantes de tipos numéricos
UPPER_BAND	Límite superior	Líneas de indicadores
UPPER_HISTOGRAM	Histograma de arriba	Líneas de indicadores

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
UPPER_LINE	Línea superior	Líneas de indicadores
USHORT_MAX	Valor máximo que puede ser representado por el tipo ushort	Constantes de tipos numéricos
VOLUME_REAL	Volumen comercial	Constantes de precio
VOLUME_TICK	Volumen de tick	Constantes de precio
WEDNESDAY	Miércoles	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
WHOLE_ARRAY	Significa el número de elementos que se quedan hasta el final del array, es decir, el array entero será procesado	Otras constantes
WRONG_VALUE	La constante puede convertirse implícitamente al tipo de	Otras constantes

© 2000-2025, MetaQuotes Ltd.

Constante	Descripción	Uso
	cualquier enumeración.	

© 2000-2025, MetaQuotes Ltd.